目 录

第 1 章　MATLAB 基础

1.1　MATLAB 的历史

1.1.1　MATLAB 的产生

MATLAB 的产生是与数学计算紧密联系在一起的。20 世纪 70 年代中期，Celve Moler 及其同事在美国国家基金会的资助下，开发了 LINPACK 和 EISPACK 的 Fortran 子程序库。70 年代后期，身为美国新墨西哥州大学计算机系主任的 Celve Moler 在给学生上线性代数课时，为了让学生能使用 LINPACK 和 EISPACK 程序库又不至于在编程上花费过多时间，为学生编写使用 LINPACK 和 EISPACK 的接口程序。他将这个接口程序取名为 MATLAB（即 Matrix 和 Laboratory 的前三位字母组合，意为“矩阵实验室”）。这个程序获得了很大的成功，受到学生的广泛欢迎。在以后的几年内 MATLAB 在许多大学使用，并作为面向大众的免费软件广为流传。

20 世纪 80 年代初期，Celve Moler 和 John Lttle 采用 C 语言编写了 MATLAB 的核心，合作开发了 MATLAB 第二代专业版，大大提高了它的运算效率。不久，他们成立了 MathWorks 公司并将 MATLAB 正式推向商业市场。

1.1.2　MATLAB 的发展

MathWorks 公司正式推出 MATLAB 以后，经过几十年的研究，不断完善 MATALB 的功能，使其在原有的基础上增加了许多功能。目前 MATLAB 已经成为国际上公认的优秀数学软件之一。现在 MATLAB 已经推出 6.5 版本，占据了数值型软件市场的主导地位。

MATLAB 在以下的领域里解决各种问题是一个十分有效的工具：

- 工业研究与开发。
- 数学教学，特别是线性代数。所有基本概念都能涉及。
- 在数值分析和科学计算方面的教学与研究。能够详细地研究和比较各种算法。
- 在诸如电子学、控制理论和物理学等工程和科学学科方面的教学与研究。
- 在诸如经济学、化学和生物学等有计算问题的所有其他领域中的教学与研究。

1.2　MATLAB 系统构成

1.2.1　MATLAB 的系统构成

MATLAB 系统由以下 5 大部分构成：



- MATLAB 语言；
- MATLAB 工作环境；
- MATLAB 数学函数库；
- MATLAB 图形处理系统；
- MATLAB 应用程序接口（API）。

下面对这五个部分进行详细的介绍。

1. MATLAB 语言

MATLAB 是一个可视化的计算程序，被广泛使用于从个人计算机到超级计算机范围内的各种计算机上。MATLAB 包括命令控制、可编程，有上百个预先定义好的命令和函数。这些函数能通过用户自定义函数进一步扩展。

2. MATLAB 工作环境

MATLAB 的工作环境是一个集成化的工作空间，它可以让用户输入输出数据，并提供了 M 文件的集成编译和调试环境。它包括命令窗口、M 文件编辑调试器、MATLAB 工作空间和在线帮助文档。

3. MATLAB 数学函数库

MATLAB 有许多强有力的数学函数，如正弦、指数运算、求解微分方程、傅立叶变换等复杂的函数。例如，MATLAB 能够用一个单一的命令求解线性系统，能够完成大量的高级矩阵处理。

4. MATLAB 图形处理系统

MATLAB 有强有力的二维、三维图形工具。MATLAB 能与其他程序一起使用。例如，MATLAB 的图形功能可以在一个 FORTRAN 程序中完成可视化计算。MATLAB 还提供了图形用户界面定制。

5. MATLAB 应用程序接口（API）

MATLAB 应用程序接口（API）是一个让 MATLAB 语言同 C、FORTRAN 等其他高级编程语言进行交互的函数库，该函数库的函数通过动态连接库（DLL）来读写 MATLAB 文件。它的主要功能包括在 MATLAB 中调用 C 和 FORTRAN 程序，以及在 MATLAB 和其他应用程序间建立客户/服务器关系。

1.2.2 MATLAB 工具箱及应用介绍

MATLAB 拥有一个专用的家族产品，用于解决不同领域的问题，例如：信号分析、系统识别和仿真等。这些所谓的工具箱都用于 MATLAB 的计算和画图。他们通常是 M 文件和高级 MATLAB 语言的集合，以使得用户可以方便地修改函数的原代码，或增加新的函数。用户还可以很方便地结合使用不同工具箱中的技术来设计针对某个问题的用户解决方案。由于 MATLAB 每年都会开发出一些新的工具箱，所以，在一般情况下，工具箱的列表不是固定不变的，目前 MATLAB 中工具箱的最新信息可以在 http://www.mathworks.com/products/看到。这些工具箱主要包括：

1. 应用数学类工具箱

- 模糊逻辑工具箱
- 优化工具箱

- 样条工具箱
- 统计工具箱
- 偏微分方程工具箱

2. 电子技术类工具箱

- 信号处理工具箱
- 小波工具箱
- 通信工具箱

3. 自动控制类工具箱

- 控制系统工具箱
- 线性矩阵不等式控制工具箱
- 频域系统辨识工具箱
- 鲁棒控制工具箱
- 模型预测控制工具箱

同时 MATLAB 还提供了许多专业的工具箱和开发包，比如 Motorola DSP Developer's Kit（Motorola DSP 开发包）等。这些工具箱和开发包为非常专业的应用提供了很大方便。

此外，MATLAB 还有一个功能强大的、可视化的、交互环境的工具 SIMULINK，用于模拟非线性动态系统。SIMULINK 提供一个用于创建动态系统对角模块的图形用户界面。由于 SIMULINK 充分利用了窗口技术，用户可以很容易地创建线性的、非线性的、离散的、连续的和混合的模型。由于点击—拖动操作和鼠标交互的使用，来自块库的组件可以相互连结使用。在做“what if”分析的过程中，可以改变参数。SIMULINK 与 MATLAB 充分集成，与 MATLAB 和 MATLAB 工具箱一起使用，用户可以在建模、设计、分析和仿真的不同阶段之间移动。SIMULINK 是可以进行扩展的。该环境中包含了可选工具集，如提高仿真速度。与 SIMULINK 相联系的工具箱成为块集，块集扩展了有专门设计和分析能力的块库。

需要了解 MATLAB 中工具箱的详细情况，可以输入 demo（以前版本可以输入 expo），然后选择 toolbox 得到。随着 MATLAB 版本的升级，工具箱也有一些改动，相应工具箱的函数或是使用方法有一定的差异，本书是基于 MATLAB6.5 的。

1.3　开始使用 MATLAB

1.3.1　MATLAB 的启动

不同的计算机系统，MATLAB 的启动也不一样。在 Windows 和 Macintosh 系统中，程序通常通过点击一个图标而启动。在 UNIX 系统中，程序是通过在命令行系统提示符后键入如下字符启动：matlab。

如果上述工作有问题，可请教系统管理员。当启动 MATLAB 时，如果 matlabrc.m 和 startup.m 文件存在，则执行这些文件。在这些文件中，为满足个人需要，用户可以给定命令以调整 MATLAB。例如，constants 用于设置图形等。在一个多用户系统上，系统管理员存储 matlabrc.m 文件，但你也能为自己的使用创建文件 startup.m。要退出 MATLAB，键入 quit 或 exit。在 Windows 2000 上打开的 MATLAB6.5 窗口如图 1-1 所示。



图 1-1　MATLAB 命令窗口

除此之外，对一些系统有指定的菜单选择。例如，在 Windows 和 Macintosh 系统中，在文件菜单下可以找到选项 quit。当编辑或执行 MATLAB 时，下列的快捷键十分有用。通常因为不同的平台使用不同的键，因此，给定了一些替换键。用户在自己的系统上试一下这些键，注意哪些键组合使用，见表 1-1。

表 1-1　特殊的功能键

快 捷 键	说　明
↑或 Ctrl_p	恢复前面的命令
↓或 Ctrl_n	恢复当前命令之后键入的命令
→或 Ctrl_f	向右移动一个字符
←或 Ctrl_b	向左移动一个字符
Delete, Backspace	删除字符
Ctrl_l 或 Ctrl_←	向左移动一个字
Ctrl_r 或 Ctrl_→	Ctrl_r 或 Ctrl_→向右移动一个字
Ctrl_a 或 Home	移动到行的第一个字符
Ctrl_e	Ctrl_e 移动到行尾
Ctrl_k	删除到行尾
cedit	cedit 在不同的快捷键间转换

1.3.2　样例

MATLAB 能用于计算，并以二维和三维图形显示各种函数。在 MATLAB 函数中包括了所有主要的数学函数和大量的高级函数。



【例1】 用简短的MATLAB命令计算并绘制在0≤x≤6范围内的sin(2x)、sin(x^2)和(sin$x$)^2。

x = linspace (0 , 6) ; % 创建一个向量x。

y1 = sin(2*x) ; % 向量y1等于x坐标上某一x的sin(2x)值。

y2 = sin(x . ^ 2) ; % 向量y2等于sin(x.^ 2)，同上。

y3 = (sin(x)). ^ 2 ; % 向量y3等于(sin(x)).^ 2，同上。

命令plot(x,y1)绘制向量y1，y1作为向量x的一个函数。由此能够很容易地在一个图上绘制sin(2x)、sin(x^2)和(sin$x$)^{2}的曲线并正确地标记它们，如图1-2所示。

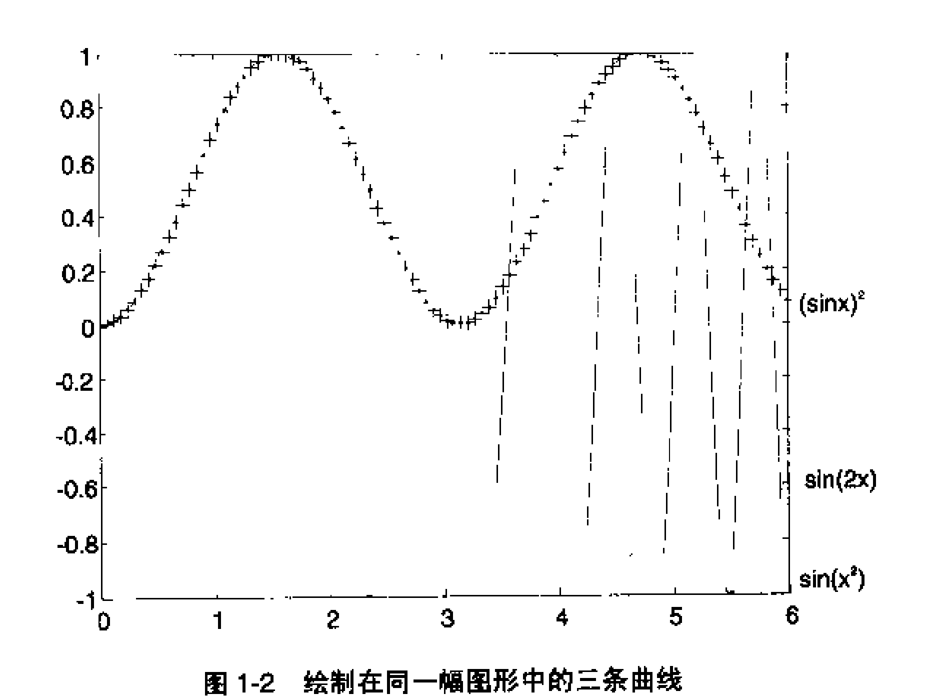


图1-2 绘制在同一幅图形中的三条曲线

【例2】 绘制三维图形。MATLAB可以很好地绘制三维图形，在图1-3中分别用surf、mesh、waterfall和contour3方法绘制图形。

程序分别如下：

(1) surf

```
k = 5;
n = 2^k-1;
[x,y,z] = sphere(n);
c = hadamard(2^k);
surf(x,y,z,c);
colormap([1  1  0; 0  1  1])
axis equal
```

(2) mesh

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5])
```

(3) waterfall

```
[X,Y,Z] = peaks(30);
waterfall(X,Y,Z)
```

(4) contour3

```
[X,Y] = meshgrid([-2:.25:2]);
Z = X.*exp(-X.^2-Y.^2);
contour3(X,Y,Z,30)
surface(X,Y,Z,'EdgeColor',[.8 .8 .8],'FaceColor','none')
grid off
view(-15,25)
colormap cool
```

图 1-3　使用四种方法绘制函数图形

【例 3】 示例程序。MATLAB 中提供了丰富的示例程序。在打开 MATLAB 后的命令提示符下输入 demo，可以进入演示界面，选择 toolbox，如图 1-4 所示。

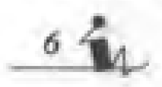




图 1-4　演示窗口

选择相应的工具箱，单击右下角的演示，可以获得对应该工具箱的演示窗口。例如运行样条工具箱的演示得到图 1-5。



图 1-5　样条曲线工具箱的演示窗口

选择窗口右侧的 Start 按钮，可以运行演示程序，如图 1-6 所示。在窗口的下方显示了该插值函数的程序，在上方的图形区则绘制了相应的图形。用户可以点击 next 按钮观察下一个演示，也可以单击 Autoplay 观察自动播放演示。

对 MATLAB 工具箱不熟悉的用户可以通过 demo 对工具箱的功能有个大概的了解。

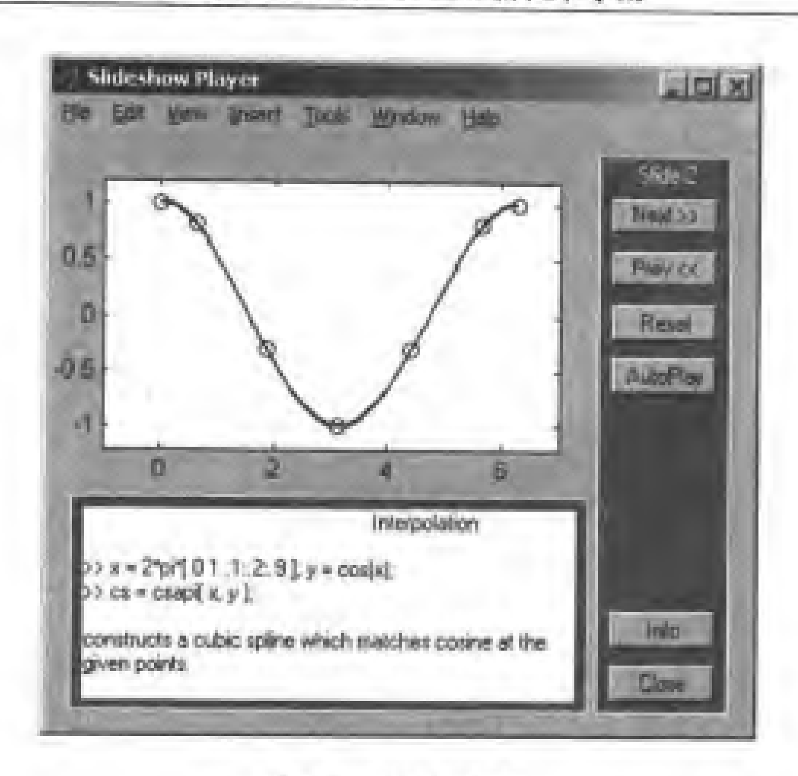


图 1-6　样条曲线演示

1.3.3　MATLAB 初步知识

通常，MATLAB 能被当作计算器使用（>>为 MATLAB 的提示符）：

```
>> 4*45+4596
ans =
     4776
```

同一行上可以有多条命令：

```
>> 2^40,sin(pi/4)
ans =
  1.0995e+012
ans =
     0.7071
```

一般来讲，变量用于保存所赋的值和结果。如果没有赋值，MATLAB 将结果存放在名为 ans 的变量中。现在定义变量并赋值：

```
>> x=238
   x =
238
>> y=x^2
   y =
    56644
```

圆括号'()'，可在数学式子中使用。

注意：在命令行尾的分号 ';' 是 MATLAB 'quietly' 执行赋值命令，即在屏幕上不回显

信息，但计算照常执行。

MATLAB 中的变量通常为向量或矩阵：

```
>>zcol=[1; 2; 3; 4; 5],zrow=[1 2 3 4 5]
zcol =
     1
     2
     3
     4
     5
zrow =
     1     2     3     4     5
>> Hm=[1 2 3;4 5 6;7 8 9;1 2 3;4 5 6;7 8 9]
Hm =
     1     2     3
     4     5     6
     7     8     9
     1     2     3
     4     5     6
     7     8     9
```

到现在为止，已经定义了许多变量。输入以下命令，可以得到变量列表：

```
>> who
Your variables are:
Hm      zcol   zrow   ans
x       y
```

MATLAB 在程序运行过程中保存所有的变量，清除变量应输入 clear。先前的变量现在被全部清除。此时，如果输入 who，将不会返回任何信息。

向量可通过使用元素操作运算符来生成：

```
>> ovector=0:6
ovector =
     0     1     2     3     4     5     6
```

使用 2 为步长：

```
>> uvector=0:2:6
uvector =
     0     2     4     6
```

通过使用双操作符向量也可直接计算向量：

```
>> uvector.^2
ans =
     0     4    16    36
```

注意：先前使用的运算符^，表示应对向量中的每一个元素进行操作。借助箭头键，能重

复先前所给的命令。如果输入有误，它就能避免再写过长的表达式，这样能节省很多时间。

将向量或矩阵放入括号中能定义一个新的表达式，但是大小必须匹配：

```
>> X=[uvector;uvector.^2;uvector.^3]
X =
     0     2     4     6
     0     4    16    36
     0     8    64   216
```

第2章 MATLAB系统辨识工具箱

控制理论的发展已经走过数十年，取得了丰富的成果，并在工程中得到了非常广泛的应用。而控制理论都是基于对控制对象的各种动力特性进行数学建模。随着控制对象的复杂化和大型化，许多情况下我们只能获得对象的输入输出的数据。因此，根据这些数据进行建模，从而了解系统本来的特性，就成了控制理论的一项重要内容。

MATLAB 的系统辨识工具箱提供了进行系统模型辨识的有力工具，其主要的功能包括：

- 非参数模型辨识工具；
- 参数模型辨识工具，包括 AR、ARX、状态空间和输入误差等模型类的辨识工具；
- 模型验证工具；
- 递推参数估计；
- 各种模型类的建立和转换函数；
- 集成多种功能的图形用户界面（GUI）。

2.1 系统辨识的原理及辨识模型的简介

2.1.1 基本原理

系统辨识的主要内容包括：

1. 实验设计

系统辨识实验设计需要完成：确定输入信号、采样周期、辨识时间和辨识模式。

例如采样周期要满足以下 3 条规则：

(1) 满足采样定理，即采样频率不能低于信号截止频率的 2 倍；

(2) 考虑辨识的算法，控制计算速度和检测元件的响应速度等；

(3) 工程中常用的采样公式为：$T_0=T_{95}/(5\sim15)$

其中 T_0 表示采样周期，T_{95} 表示过程阶跃响应达到稳态值 95%所需要的调节时间。

2. 模型结构辨识

模型结构辨识包括模型类和模型结构参数确定。模型类确定主要是根据经验和对实际模型特性的一些了解，决定模型的类型。确定模型后可以根据输入输出数据决定模型中所用的参数。

3. 模型参数辨识

确定使用的模型后，需要进行模型的参数辨识。如使用最小二乘法等。

4. 模型的检验

在完成模型的结构和参数的辨识后，需要对模型进行检验，这是必不可少的步骤之一。检验的方法有：

(1) 采用不同时间内采集的数据，分别对模型进行辨识，如果结果得到的模型基本相同，则说明辨识是可靠的。

(2) 取两组数据进行辨识，并计算它们的损失函数，再交叉使用两组数据，再计算分别的损失函数，如果损失函数没有很大的变化，则说明模型辨识是可靠的。

(3) 检验模型与对象输出的残差{e(k)}的白色性，如果残差序列可以看作零均值的白色噪声序列，则可以认为模型辨识是可靠的。

(4) 增加辨识中使用的数据长度，如果损失函数不再明显的下降，则可以认为模型辨识是可靠的。

2.1.2 常用的模型类

1. 非参数模型类

在非参数模型类中主要包括脉冲响应模型和频域描述模型。

如图 2-1 所示，假设方框中的系统为线性系统，u 为输入，y 为输出，v 为噪声，则可以得出输入输出的关系如下：

$$y(t)=G(q)u(t)+v(t)$$

其中，q 为移位算子；$G(q)u(t)$是一种简写形式。

$$G(q)u(t)=\sum_{k=1}^{\infty}g(k)u(t-k)$$

$$G(q)=\sum_{k=1}^{\infty}g(k)q^{-k}\;;\qquad q^{-1}u(t)=u(t-1)$$

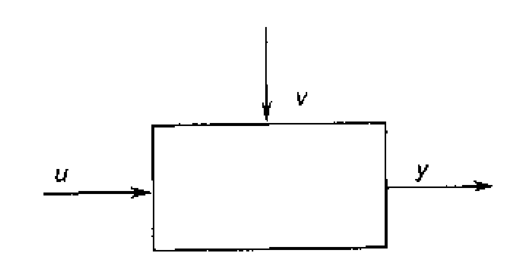


图 2-1 线性对象

其中，q 为时间平移算子。序列$\{g(k)\}$称为对象的脉冲响应模型。$v(t)$是不可测量的附加干扰(噪声)。它的特性可以用它的(自)频谱来表示

$$\Phi_v(\omega)=\sum_{\tau=-\infty}^{\infty}R_v(\tau)\mathrm{e}^{-i\omega\tau}$$

$$R_v(\tau)=Ev(t)v(t-\tau)$$

这两种模型类都不能表示为对象的有限参数模型形式，因此称为非参数模型。

2. 参数模型类

参数模型类实质利用有限的参数来表示对象的模型，在系统辨识工具箱中支持的参数模

型有 ARX 模型，ARMAX 模型，BJ 模型，输入误差模型和状态空间模型。通常都限定为以下特殊的情形：

(1) ARX 模型。ARX 模型具有如下形式：

$$A(q)y(t)=B(q)u(t-nk)+e(t)$$

(2) ARMAX 模型。ARMAX 模型具有如下形式：

$$A(q)y(t)=B(q)u(t-nk)+C(q)e(t)$$

(3) BJ 模型。BJ 模型（Box-Jenkins 模型）具有如下形式：

$$A(q)y(t)=[B(q)/F(q)]u(t-nk)+[C(q)/D(q)]e(t)$$

(4) 输入误差模型（Output-Error）。

$$y(t)=[B(q)/F(q)]u(t-nk)+e(t)$$

(5) 状态空间模型。状态空间模型适于多变量系统，其形式如下：

$$x(t+1)=Ax(t)+Bu(t)$$

$$y(t)=Cx(t)+Du(t)+v(t)$$

其中，A, B, C 和 D 为状态空间模型的系数矩阵，$v(t)$为外界噪声信号。

2.2　系统辨识工具箱函数

2.2.1　模型建立和转换的函数介绍

在 MATLAB 的系统辨识工具箱中提供了许多模型建立函数，见表 2-1。

表 2-1　模型建立函数

函数名	功　能
iddata	标准包含输入输出数据的对象
idmodel	基本的模型对象，综合了许多模型的公共特点
idarx	从 ARX 多项式建立 ARX 模型
idgrey	根据 M 文件定义 idgrey 模型
idpoly	构造基于输入输出模型的 idpoly 模型
idss	构造状态空间模型
idfrd	构造 idfrd 模型
init	设置模型的参数

1. iddata

功能：iddata 是工具箱中处理信号的一个最基本的对象，在以下的很多函数中都要到。

语法：data = iddata(y,u)

data=iddata(y,u,Ts,'P1',V1,...,'PN',VN)

说明：y 是一列向量或 N×ny 的矩阵，y 的列对应于不同的输出频道，类似的 u 是一列向量或 N×ny 的矩阵，u 的列对应于不同的输入频道。data = iddata(y,u,Ts)生成一个包含这些输

入输出频道的 iddata 对象，Ts 是采样区间。对于大部分的应用来讲该构造已经足够了。对于时间序列（没有输入信号），可以使用 data=iddata(y),或令 u=[]。同样 iddata 对象也可以用于单输入系统，只需令 y=[]。

2. idmodel

idmodel 对象综合了所有模型如 idarx, idgrey, idpoly 和 idss 的公共特点。所有参数估计都返回 idmodel 对象。

如果要建立一个模型，需要以下的模型建立函数：

- idarx：建立多变量的 ARX 模型；
- idgrey：用户定义的灰箱状态空间模型；
- idpoly：单输出的多项式模型；
- idss：状态空间模型。

下面介绍这几个模型建立函数：

3. idarx

功能：从 ARX 多项式建立 ARX 模型。

语法：m = idarx(A,B,Ts)

m = idarx(A,B,Ts,'Property1',Value1,...,'PropertyN',ValueN)

说明：多输入输出的 ARX 模型有如下的形式：

$$y(t)+A_1 y(t-1)+A_2 y(t-2)+\cdots+A_{na} y(t-na)=$$

$$B_0 u(t)+B_1 u(t-1)+\cdots+B_{nb} u(t-nb)+e(t)$$

其中系数 A_k 为 $ny\times ny$ 维矩阵，B_k 为 $ny\times nu$ 维矩阵（ny 为输出参数个数，nu 为输入参数个数）

输入参数 A 为 $ny\times ny*(na+1)$ 维的矩阵使得：

A(:,:,k+1) = A_k

A(:,:,1) = eye(ny)

B 为 ny×nu*(nb+1)维的矩阵使得：

B(:,:,k+1) = B_k;

参数 Ts 为采样周期；

【例 1】 模拟一个具有 1 输入 2 输出的二阶 ARX 模型并使用模拟数据估计该对象。

```
A = zeros(2,2,3);
B = zeros(2,1,3);
A(:,:,1) =eye(2);
A(:,:,2) = [-1.5 0.1;-0.2 1.5];
A(:,:,3) = [0.7 -0.3;0.1 0.7];
B(:,:,2) = [1;-1];
B(:,:,3) = [0.5;1.2];
m0 = idarx(A,B,1);
u = iddata([],idinput(300));
e = iddata([],randn(300,2));
```

```
y = sim(m0,[u e]);
m = arx([y u],[[2 2;2 2],[2;2],[1;1]]);
```

4. idgrey

功能：由用户的 M 文件定义的灰箱模型结构生成 idgrey 模型。

语法：m = idgrey(MfileName,ParameterVector,CDmfile)

m = idgrey(MfileName,ParameterVector,CDmfile,FileArgument,Ts,'Property1', Value1,..., 'PropertyN',ValueN)

说明：MfileName 为 M 文件的文件名。该文件描述了状态空间矩阵如何依赖于被估计的参数。M 文件的格式为：

[A,B,C,D,K,X0] = mymfile(pars,Tsm,Auxarg)

ParameterVector：为列向量，其长度必须等于模型中自由参数的个数。

Cdmfile：描述用户编写的 M 文件如何处理连续/离散时间模型。它可以取如下的值：

CDmfile = 'cd'：当参数 Tsm=0 时，返回连续时间状态空间模型；当参数 Tsm>0 时，返回离散时间状态空间模型，采样周期为 Tsm，此时 M 文件必须包括采样的程序：

- CDmfile = 'c'：M 文件始终返回连续时间状态空间模型；
- CDmfile = 'd'：M 文件始终返回离散时间状态空间模型；

【例 2】 先编写 M 文件如下：

```
function [A,B,C,D,K,x0] = mynoise(par,T,aux)
R2 = aux(1);
A = [par(1) par(2);1 0];
B = [1;0];
C = [par(3) par(4)];
D = 0;
R1 = [par(5) 0;0 0];
K = A*dlqe(A,eye(2),C,R1,R2);
x0 = [0;0];
```

将此文件存为：mynoise.m

使用 idgrey 函数：

```
mn = idgrey('mynoise',[0.1,-2,1,3,0.2]','d')
m = pem(z,mn)
```

5. idpoly

功能：构造基于输入输出模型的 idpoly 模型。

语法：m = idpoly(A,B)

m = idpoly(A,B,C,D,F,NoiseVariance,Ts)

m = idpoly(A,B,C,D,F,NoiseVariance,Ts,'Property1',Value1,'PropertyN',ValueN)

说明：多输入单输出的模型具有如下的格式：

$$A(q)y(t)=\frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

输入参数 A，B，C，D，F 分别为多项式 $A(q)$，$B(q)$，$C(q)$，$D(q)$，$F(q)$的系数矩阵。

如果为单输入系统上述的参数则均为向量，且 A，C，D，F 的第一个元素均为 1；B 则包含 0 元素以表示系统纯时延的大小；对于多输入系统，B 和 F 均为矩阵形式，每一行对应一个输入的多项式系数；对于时间序列 B 和 F 为空矩阵 $B = []$，$F = []$。

NoiseVariance：白噪声序列的方差；

Ts：采样周期 $Ts \leq 0$ 表示连续时间系统。

【例 3】 建立一个 ARMAX 系统：

```
A = [1 -1.5 0.7];
B = [0 1 0.5];
C = [1 -1 0.2];
m0 = idpoly(A,B,C);
```

6. idss

功能：构造状态空间模型。

语法：m = idss(A,B,C,D)

m = idss(A,B,C,D,K,x0,Ts,'Property1',Value1,...,'PropertyN',ValueN)

mss = idss(m1)

说明：该函数定义具有如下形式的状态空间模型结构：

$$\tilde{x}(t) = A(\theta)x(t) + B(\theta)u(t) + K(\theta)e(t)$$

$$x(0) = x_0(\theta)$$

$$y(t) = C(\theta)x(t) + D(\theta)u(t) + e(t)$$

输入参数定义：A，B，C，D，K，$X0$ 为状态空间模型矩阵。

Ts 为采样周期，Ts=0 表示连续时间模型。

ml：给定的 idmodel 模型或 LTI 系统。

【例 4】 估计一个系统的自由参数：

```
A = [-0.2, 0; 0, -0.3]; B = [2;4]; C=[1, 1]; D = 0
m0 = idss(A,B,C,D);
m0.As = [NaN,0;0,NaN];
m0.Bs = [NaN;NaN];
m0.Cs = [1,1];
m0.Ts = 0;
m = pem(z,m0);
```

7. idfrd

功能：构造 idfrd（Identified Frequency Response Data）模型。

语法：h = idfrd(Response,Freqs,Ts)

h = idfrd(Response,Freqs,Ts,'CovarianceData',Covariance, 'SpectrumData',Spec, 'NoiseCovariance',Speccov,'Property1', Value1,'PropertyN',ValueN)

h = idfrd(mod)

h = idfrd(mod,Freqs)

说明：输入参数说明：

Response：为三维 ny×nu×Nf 的阵列，ny 为输出变量个数，nu 为输入变量个数，Nf 为频率点个数，即 freqs 的长度。Response(ky,ku,kf) 为从输入 ku 到输出 ky 在频率 Freqs(kf)处的复值频率响应。当为 SISO 系统时，Response 可以为一向量。

Freqs：包含响应频率的长度为 Nf 的列向量。

Ts：采样时间，Ts=0 表示连续时间系统。

Covariance：为 5 维 ny×nu×Nf×2×2 的阵列，ny 为输出变量个数，nu 为输入变量个数，Nf 为频率点个数。

Covariance(ky,ku,kf,:,:) 为 Response(ky,ku,kf) 的 2×2 协方差矩阵。

spec：可选参数，为 ny×ny×Nf 阵列，spec(ky1,ky2,kf) 表示输出 ky1 和 ky2 噪声间在 Freqs(kf)处的交叉频谱；

speccov：ny×ny×Nf 阵列，频谱矩阵的方差。

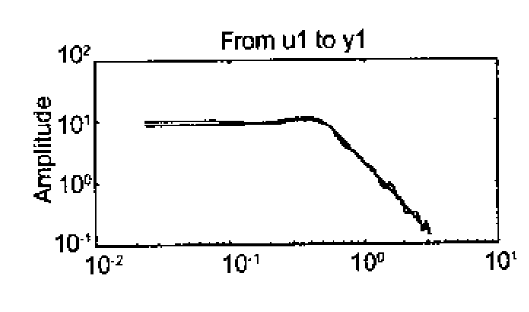
mod：给定的模型对象，用以从中构造 idfrd 模型。

在离散情况下频率缺省为：Freqs = [1:128]'/128*pi/Ts，这里的 Ts 为 mod 中的采样时间。

在连续情况下频率缺省为：Freqs = logspace(log10(pi/Ts/100),log10(10*pi/ Ts),128)，其中 Ts 为需要估计模型数据的采样时间。

【例 5】 和 ARMAX 模型比较。结果如图 2-2 所示。

```
A = [1  -1.5  0.7];
B = [0 1 0.5];
m0 = idpoly(A,B);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
m = armax(z,[2 2 2 1]);
g = spa(z)
 bode(g,m)
```



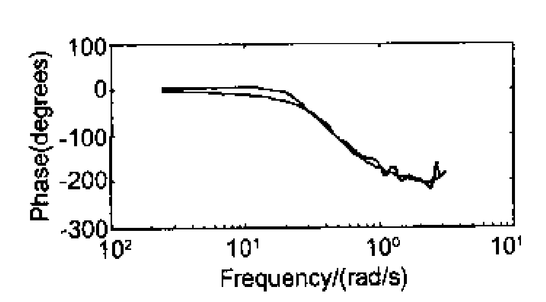


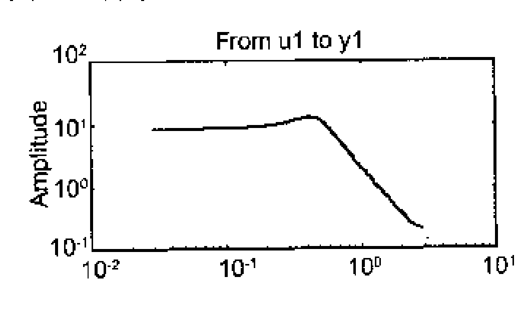
图 2-2 与 ARMAX 模型比较

构造离散的 idfrd 模型一个包含 g 另一个包含噪声频谱：

```
A = [1  -1.5  0.7];
B = [0  1  0.5];
m0 = idpoly(A,B);
```

```
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
m = armax(z,[2 2 2 1]);
g = idfrd(m('m'))
phi = idfrd(m('n'))
bode(g,m)
```

结果如图 2-3 所示。



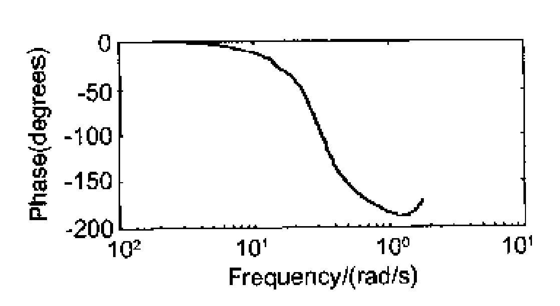


图 2-3 idfrd 对象 bode 图

8. init

功能：设置模型的参数。

语法：m = init(m0)

m = init(m0,R,pars,sp)

说明：该函数随机初始化 idmodel 对象 m0，返回对象 m 具有与 m0 相同的模型结构。

m = init(m0,R,pars,sp)：给出的参数将在参数 pars 的周围随机化，随机的半径由向量 R 给出：pars(k) + e*sqrt(R(k))，e 为标准随机数，平均为 0，半径为 1。

sp = 'b'：只允许模型稳定并且具有稳定预测器；

sp = 's'：只需要模型稳定；

sp = 'p'：缺省值，只需要模型具有稳定预测器。

模型的转换函数可以实现多种模型间的相互转换，模型结构函数可以从模型中提取需要的数据，见表 2-2。

表 2-2 模型转换和模型结构函数

函 数	功 能
c2d	将连续时间模型转换为离散时间模型
d2c	将离散时间模型转换为连续时间模型
tfdata	将模型转换为传递函数
zpkdata	计算模型的零点、极点和稳定增益
ssdata	将模型转换为状态空间模型
idmodred	对模型降阶（需要控制系统工具箱）
arxdata	从模型中提取 ARX 模型参数
freqresp	计算模型的频率函数
ss, tf, zpk, frd	将系统辨识工具箱中模型对象转换为控制工程工具箱中的 LTI 模型

(1) c2d

功能：将连续时间模型转换为离散时间模型。

语法：md = c2d(mc,T)

md = c2d(mc,T,method)

说明：mc 是任意连续时间模型，T 为采样周期。

md 为采用 T 时间采样后的离散时间模型。

method = 'zoh'：假设在采样的时间间隔内输入数据是局部常值的。

method = 'foh'：假设在采样的时间间隔内输入数据是局部线性的。

【例 6】 mc = idpoly(1,1,1,1,[1 1 0],'Ts',0);

md = c2d(mc,0.5);

pzmap(md)

结果输出如图 2-4 所示。

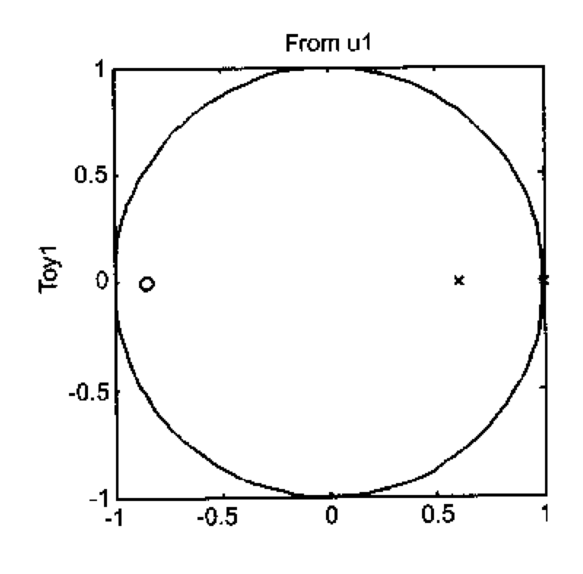


图 2-4　对象零极点图

(2) d2c

功能：将离散时间模型转换为连续时间模型。

语法：mc = d2c(md)

mc = d2c(md,'CovarianceMatrix',cov,'InputDelay',inpd)

说明：md 是离散时间模型。mc 为返回的连续时间模型。

参数'CovarianceMatrix' 用来决定是否允许协方差矩阵转换，在参数中加入'Covariance Matrix' 即可禁止转换。参数 'InputDelay' 决定是否对时延进行逼近，如许逼近则在参数中加入'InputDelay' 即可。

【例 7】 将一个已辨识的离散模型转换为连续时间模型并比较两个模型的频率响应。

m = n4sid(data,3)

mc = d2c(m);

bode(m.mc,'sd',3)

(3) tfdata

功能：将模型转换为传递函数。

语法：[num,den] = tfdata(m)

[num,den,sdnum,sdden] = tfdata(m)

[num,den,sdnum,sdden] = tfdata(m,'v')

说明：m 为任何 idmodel 对象，ny 为输出频道，nu 为输入频道。输出参数 num 为 ny×nu 的矩阵，num{ky,ku}(注意:为大括号)包含从输入 ku 到输出 ky 的传递函数的分子多项式系数构成的矩阵。

类似的 den 为对应传递函数的分母多项式系数构成的矩阵。

sdunm、sdden 和 num、den 的形式类似，它们分别包含传递函数分子分母的导数的系数矩阵。

(4) zpkdata

功能：计算模型的零点、极点和稳定增益。

语法：[z,p,k] = zpkdata(m)

[z,p,k,dz,dp,dk] = zpkdata(m)

[z,p,k,dz,dp,dk] = zpkdata(m,'v')

说明：m 为任何 idmodel 对象，ny 为输出频道，nu 为输入频道。输出参数 z 为 ny×nu 的矩阵，z{ky,ku}包含从输入 ku 到输出 ky 的转换函数零点，为列向量的形式，其元素可能为复数。类似的 p 为 ny×nu 的矩阵，它包含转换函数的极点。

k 为 ny×nu 维的矩阵，为对象的稳定增益矩阵。

dz 包含零点的协方差矩阵；

dp 包含极点的协方差矩阵；

dk 为一矩阵包含 k 中元素的方差。

(5) ssdata

功能：将模型转换为状态空间模型。

语法：[A,B,C,D,K,X0] = ssdata(m)

[A,B,C,D,K,X0,dA,dB,dC,dD,dK,dX0] = ssdata(m)

说明：m 为给定的 idmodel 的对象。A，B，C，D，K，X0 为对应状态空间模型的系数矩阵。状态空间模型具有如下的形式：

$$\tilde{x}(t) = Ax(t) + Bu(t) + Ke(t)$$

$$x(0) = x0$$

$$y(t) = Cx(t) + Dx(t) + e(t)$$

输出参数 dA，dB，dC，dD，dK，dX0 分别对应参数 A，B，C，D，K，$X0$ 的标准差。

【例 8】

```
mc = idpoly(1,1,1,1,[1 1 0],'Ts',0);
ssdata(mc)
```

MATLAB 返回：

```
ans =
    -1    1
```

0 0

(6) idmodred

功能：对模型降阶(需要控制系统工具箱)。

语法：MRED = idmodred(M)

MRED = idmodred(M,ORDER,'DisturbanceModel','None')

说明：M 为给定的 idmodel 对象，降阶后的 MRED 为 idss 模型。该函数需要控制系统工具箱中的一些函数，因此必须安装了控制系统工具箱后，才能使用该函数。

(7) arxdata

功能：从模型中提取 ARX 模型参数。

语法：[A,B] = arxdata(m)

[A,B,dA,dB] = arxdata(m)

说明：m 为 idpoly 或 idarx 模型对象。arxdata 函数可以用在任何 idarx 模型上，对 idpoly 模型必须要求 nc=nd=nf=0。

ARX 模型的格式为：

$$y(t)+A_1y(t-1)+A_2y(t-2)+\cdots+A_{na}y(t-na)=$$
$$B_0u(t)+B_1u(t-1)+\cdots+B_{nb}u(t-nb)+e(t)$$

对应的输出参数矩阵 A 为 ny*nu*(na+1)维的矩阵。

A(:,:,k+1) = Ak

A(:,:,1) = eye(ny);

相应的 B 为 $ny*nu*(nb+1)$维的矩阵。

B(:,:,k+1) = Bk;

dA 和 dB 分别为 A 和 B 的标准差。

(8) freqresp

功能：计算模型的频率函数。

语法：H = freqresp(m)

[H,w,covH] = freqresp(m,w)

说明：m 为任意 idmodel 或 idfrd 对象。

H = freqresp(m,w)计算 idmodel 对象 m 在频率 w 处的频率响应 H。

[H,w,covH] = freqresp(m,w)同时返回频率向量 w 和协方差矩阵 covH。

(9) ss, tf, zpk, frd

功能：将系统辨识工具箱中模型对象转换为控制工程工具箱中的 LTI 模型。

语法：sys = ss(mod)

sys = tf(mod)

sys = zpk(mod)

sys = frd(mod)

说明：mod 为任意 idgrey，idarx，idpoly，idss，idmodel 或 idfrd 模型，idfrd 模型只能转换为 frd 模型。

sys 作为 LTI 模型返。

2.2.2 非参数模型类的辨识函数介绍

非参数模型辨识包括脉冲响应模型和频域描述模型，这两种模型的辨识函数见表 2-3。

表 2-3 非参数模型辨识函数

函 数	功能说明
covf	估计时间序列的协方差函数
cra	采用相关分析方法估计对象脉冲响应和方差函数
etfe	直接基于快速 Fourier 变换估计对象的频率响应
spa	利用频谱分析方法估计对象的频率响应和噪声频谱

1. covf

功能：估计时间序列的协方差函数

语法：R = covf(data,M)

R = covf(data,M,maxsize)

说明：参数的定义：

data：iddata 对象；

M：计算协方差函数和互协方差函数的最大延迟；

Maxsize：可选参数，用于控制计算所需的储存容量；

R：n 维时间序列协方差和互协方差函数的估计结果，为 n^2M 维矩阵，其元素为：

$$R(i+(j-1)nz,k+1)=\frac{1}{N}\sum_{t=1}^{N}z_f(t)z_j(t+k)=\hat{R}_{ff}(k)$$

2. cra

功能：采用相关分析方法估计对象脉冲响应和方差函数

语法：cra(data);

[ir,R,cl] = cra(data,M,na,plot);

cra(R);

说明：该函数用于单输入输出对象的相关分析。

Data: iddata 对象；

M：指定计算协方差函数的最大延迟，即计算协方差函数计算的时间范围为：[-M，M]，缺省为 20;

na：白化滤波器的价次；

plot：指定是否绘制脉冲响应曲线或协方差函数曲线。为 0 时不绘制，为 1 时绘制对象的脉冲响应曲线，为 2 时绘制对象的输入输出自相关曲线；

ir：对象脉冲响应的估计；

R：输入输出的协方差函数矩阵；

cl：具有 99%信念因子的脉冲响应估计。

【例 1】 用 MATLAB 程序分别对一个二价对象进行 ARX 建模和相关分析，并比较两种方法得到的脉冲响应和阶跃响应，如图 2-5 所示。

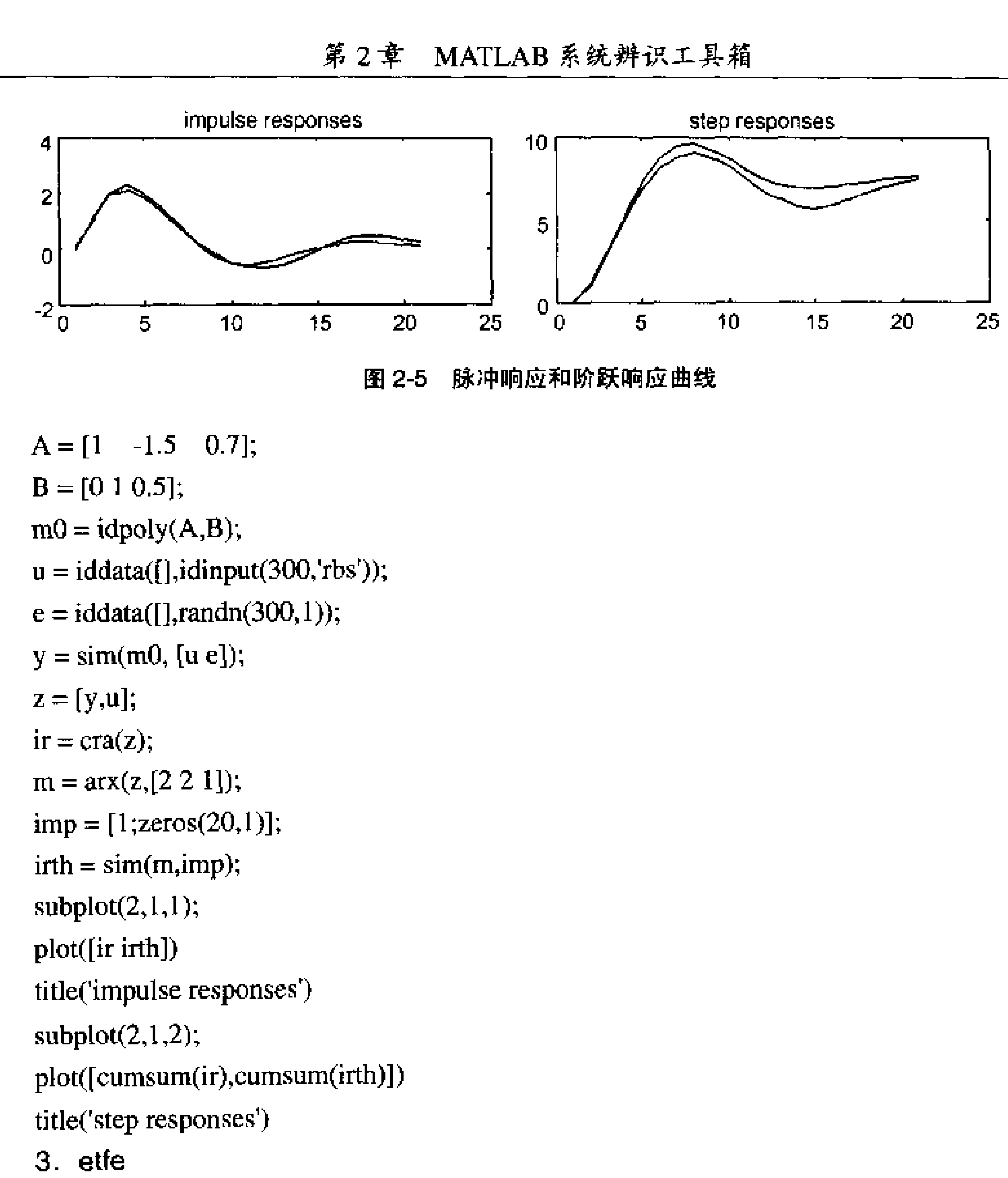


图 2-5　脉冲响应和阶跃响应曲线

```
A = [1   -1.5   0.7];
B = [0 1 0.5];
m0 = idpoly(A,B);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
ir = cra(z);
m = arx(z,[2 2 1]);
imp = [1;zeros(20,1)];
irth = sim(m,imp);
subplot(2,1,1);
plot([ir irth])
title('impulse responses')
subplot(2,1,2);
plot([cumsum(ir),cumsum(irth)])
title('step responses')
```

3．etfe

功能：直接基于快速 Fourier 变换估计对象的频率响应。

语法：g = etfe(data)

　　　g = etfe(data,M,N)

说明：data：iddata 对象；

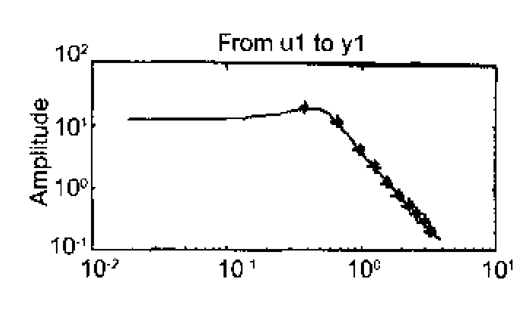
M：可选参数，指定对品扑估计的平滑操作；

N，T：可选参数，指定对计算频率响应的频率向量；

g：对象的频率响应函数。

【例 2】 生成一个周期输入，进行模拟，估计对象的频率响应，如图 2-6 所示。

```
m = idpoly([1 -1.5 0.7],[0 1 0.5]);
u = iddata([],idinput([50,1,10],'sine'));
u.Period = 50;
y = sim(m,u);
me = etfe([y u])
```



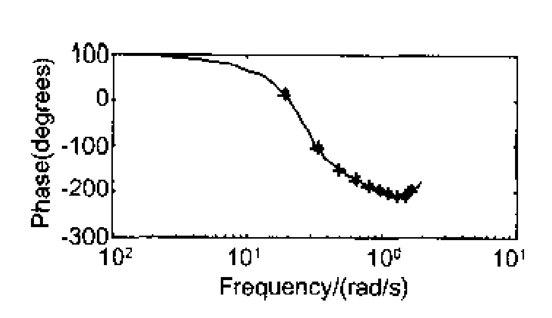


图 2-6　对象频率响应曲线

```
bode(me,'b*',m)
```

4．spa

功能：利用频谱分析方法估计对象的频率响应和噪声频谱。

语法：g = spa(data)

　　　g = spa(data,M,w,maxsize)

　　　[g,phi,spe] = spa(data)

说明：输入参数定义为：

data：iddata 对象；

M：可选参数，滞后窗的宽度，缺省值为 M = min(30,length(data)/10)；

W：可选参数，为频率向量形式，用于指定计算频谱函数的频率点，缺省值为 w = [1:128]/128*pi/Ts

maxsize：可选参数，用于计算过程中的存储与速度控制。

输出参数定义为：

g：对象的频率响应函数估计；

phi：噪声的频谱估计。

【例 3】

```
 A = [1   -1.5   0.7];
B = [0 1 0.5];
m0 = idpoly(A,B);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
w = logspace(-2,pi,128);
g= spa(z,[],w);
bode(g,3)
bode(g('noise'),3)
```

对象的噪声频谱估计的波特图，如图 2-7 所示。

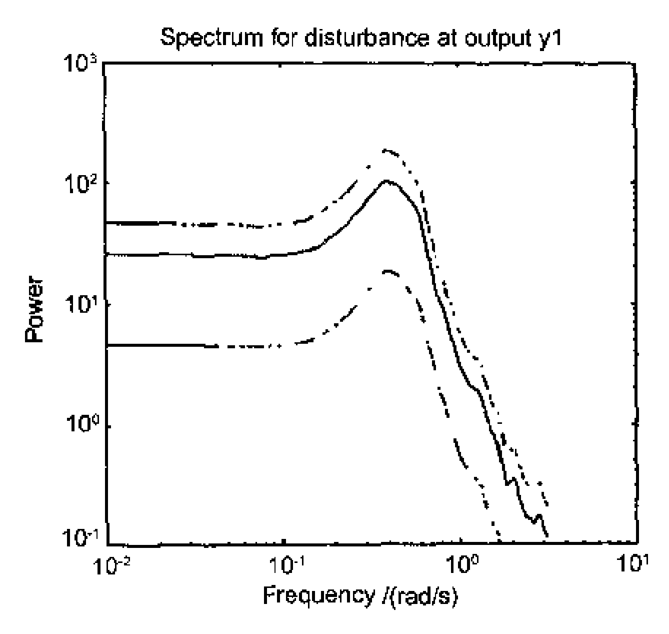


图 2-7　对象的噪声频谱曲线

2.2.3　参数模型类的辨识函数介绍

MALTAB 系统辨识工具箱支持的参数模型包括：AR/ARX、ARMAX、BJ、SS 和 OE 模型等。在系统辨识工具箱中提供了相应的模型辨识函数，见表 2-4。

表 2-4　参数模型辨识函数

函　数	功　能　说　明
Ar	时间序列的 AR 模型辨识
Ivar	基于最优辅助变量选择的 AR 模型辨识
Arx	基于最小二乘法估计的 ARX 模型辨识
iv4	采用近似最优 4 阶段辅助变量方法的 ARX 模型辨识
Ivx	采用任意辅助变量的 ARX 模型辨识
Armax	估计 ARMAX 或 ARMA 模型的参数
Oe	基于预测误差方法的输出误差模型辨识
Bj	基于预测误差方法的 BJ 模型辨识
n4sid	基于子空间方法的状态空间模型辨识
Pem	采用预测误差算法辨识一般线性输入输出模型

1．ar

功能：时间序列的 AR 模型辨识。

语法：m = ar(y,n)

[m ,refl] = ar(y,n,approach,window,maxsize)

说明：输入参数定义为

y：对象在白噪声作用下的输出；

n：AR 模型的阶次 na；

(1) approach：用于指定参数估计的最小二乘类方法，approach 的取值可为：

• 'fb'：前向-后向方法为缺省方法，该方法以前向模型和后向模型的预测误差平方和为参数估计的极小化指标；

• 'ls'：标准的最小二乘方估计方法，以前向模型的预测误差平方和为极小化指标；

• 'yw'：采用 YULE-WALKER 方法进行参数估计，该方法通过求解由采样方差构成的 YULE-WALKER 方程获得参数的估计；

• 'burg'：采用 BURG 的基于网格的方法，该方法利用前向和后向平方预测误差的一致均值求解网格滤波器方程以得到参数的估计；

• 'gl'：采用几何网格方法进行参数估计，该方法利用前向和后向平方预测误差的几何均值，求解网格滤波器方程以得到参数的估计；

• window：选择数据窗口的类型，window 的取值由以下几种定义：

• 'now'：无数据窗口，为缺省值；

• 'prw'：采用前向窗口，即测量开始点之前的 n 个输出数据被置 0，极小化指标的求和下限为 0 时刻，n 为对象 AR 模型阶次；

• 'pow'：采用后向窗口，即测量结束时刻后的 n 个输出数据点被置 0，极小化指标的求和上限为 N+n，N 为观测点个数，n 为对象 AR 模型阶次；

• 'ppw'：同时采用前向和后向窗口，即测量开始点之前的 n 个数据和测量结束时刻后的 n 个输出数据点均被置 0，这种数据窗口用于 YULE-WALKER 参数估计方法。

(2) maxsize：用于指定计算过程中允许生成的最大维数，缺省值的设定在文件 idmsize 中完成。

输出参数定义为：

m：AR 模型对应的 idpoly 对象格式；

refl：当采用基于网格的方法是，refl 用于返回反射系数和对应的损失函数。

【例 1】 对时间序列 y(t)=sin(t)+0.5*e(t)进行 AR 模型估计，并绘制频率响应波特图，其中采用缺省的参数估计方法的 AR 模型频谱，如图 2-8 所示。

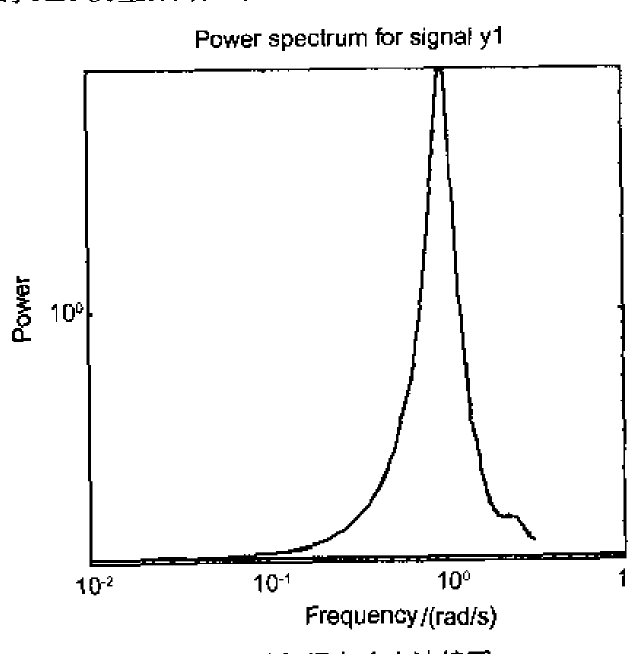


图 2-8 对象频率响应波特图

```
y = sin([1:300]') + 0.5*randn(300,1);
y = iddata(y);
mb = ar(y,4,'burg');
mfb = ar(y,4);
bode(mb,mfb)
```

2. ivar

功能：基于最优辅助变量选择的 AR 模型辨识。

语法：m = ivar(y,na)

m = ivar(y,na,nc,maxsize)

说明：本函数对如下的具有有色噪声输入的 AR 模型进行辨识

$$A(q)y(t) = v(t)$$

其中，$v(t)$为有色噪声，并且在参数估计过程中被假定阶次为 nc 的滑动平均过程。

输入参数的定义为：

y：列向量格式的输入序列；

na：指定 AR 模型的阶次；

nc：指定噪声特性的阶次，缺省为 nc=na；

maxsize：用于指定计算过程中允许生成的最大维数，缺省值的设定在文件 idmsize 中完成。

输出参数的定义为：

m：AR 模型对应的 idpoly 格式。

【例 2】利用最小二乘法和辅助变量法对过程 y(t)=sin(1.2t)+sin(1.5t)+0.2v 进行 AR 模型辨识，如图 2-9 所示。

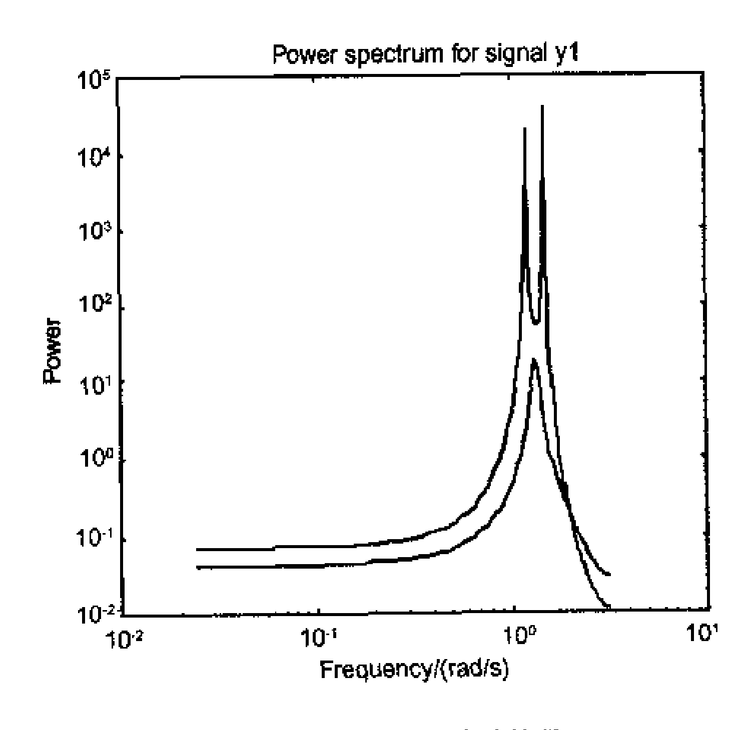


图 2-9　对象频率响应波特图

```
y = iddata(sin([1:500]'*1.2) + sin([1:500]'*1.5) +0.2*randn(500,1),[]);
miv = ivar(y,4);
mls = ar(y,4);
bode(miv,mls)
```

3．arx

功能：基于最小二乘法估计的 ARX 模型辨识。

语法：m = arx(data,orders)

m = arx(data,'na',na,'nb',nb,'nk',nk)

m= arx(data,orders,'Property1',Value1,...,'PropertyN',ValueN)

说明：data：包含输入输出数据的基础 iddata 对象；

order：指定 ARX 模型的阶次和纯时延大小，orders = [na nb nk]特别的有：

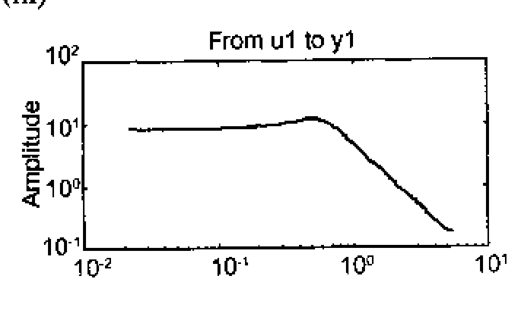
$$na:\quad A(q)=1+a_1q^{-1}+\cdots+a_{na}q^{-na}$$

$$nb:\quad B(q)=b_1+b_2q^{-1}+\cdots+b_{nb}q^{-nb+1}$$

m：对于信号输出数据，此值为 idploy 对象；对于其他的为 idarx 对象。

【例 3】 输出结果如图 2-10 所示。

```
A = [1  -1.5  0.7]; B = [0  1  0.5];
m0 = idpoly(A,B);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
m = arx(z,[2 2 1]);
bode(m)
```



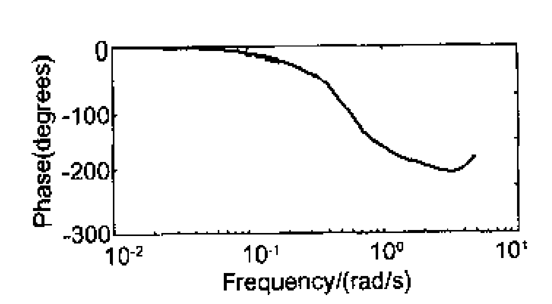


图 2-10 对象频率波特图

4．iv4

功能：采用近似最优 4 阶段辅助变量方法的 ARX 模型辨识。

语法：m = iv4(data,orders)

m = iv4(data,'na',na,'nb',nb,'nk',nk)

m= iv4(data,orders,'Property1',Value1,...,'PropertyN',ValueN)

说明：该函数和 ARX 类似，参数的定义和 ARX 也类似，它们的惟一区别是该函数对模型方程中噪声项的色不敏感。



【例 4】

```
z = iddata(y, [u1 u2]);
nb = [2 2];
nk = [0 2];
m= iv4(z,[2 nb nk]);
```

5. ivx

功能：采用任意辅助变量的 ARX 模型辨识。

语法：m = ivx(data,orders,x)

m = ivx(data,orders,x,maxsize)

说明：该函数的输入输出变量见 ARX 的定义。

x 为辅助变量矩阵。

ARMAX 模型方程具有如下形式：

$$A(q)y(t) = B(q)\,u\,(t - nk) + C(q)e(t)$$

6. armax

功能：估计 ARMAX 或 ARMA 模型的参数。

语法：m = armax(data,orders)

m = armax(data,'na',na,'nb',nb,'nc',nc,'nk',nk)

m = armax(data,orders,'Property1',Value1,...,'PropertyN',ValueN)

说明：armax 返回的值 m 是一个 idpoly 对象。用于完成基于预测误差的 ARMAX 模型辨识。

data：包含输入输出数据的 iddata 对象；

模型的次数可以简单的写作：(...,'na',na,'nb',nb,...)或设参数 order 为 orders = [na nb nc nk]

参数 na 为 AR 部分的阶次，nb 为 MA 部分的阶次，nc 为噪声特性的阶次，nk 为对象的纯时延；特别的有：

$$na:\quad A(q) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}$$

$$nb:\quad B(q) = b_1 + b_2 q^{-1} + \cdots + b_{nb} q^{-nb+1}$$

$$nc:\quad C(q) = 1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}$$

7. oe

功能：基于预测误差方法的输出误差模型辨识。

语法：m = oe(data,orders)

m = oe(data,'nb',nb,'nf',nf,'nk',nk)

m = oe(data,orders,'Property1',Value1,'Property2',Value2,...)

说明：oe 函数使用预测误差的方法估计输出误差模型：

$$y(t) = \frac{B(q)}{F(q)}u(t - nk) + e(t)$$

data 是包括输出输入数据的基本 iddata 对象。

系统结构的信息可以简单写为：(...,'nb',nb,'nf',nf,'nk',nk,...)或者利用参数 order 定义：orders = [nb nf nk]，参数 nb、nf 为输出误差模型的次数，nk 为时延。特别的有：

$$nb:\quad B(q)=b_1+b_2q^{-1}+\cdots+b_{nb}q^{-nb+1}$$

$$nf:\quad F(q)=1+f_1q^{-1}+\cdots+f_{nf}q^{-nf}$$

常用的 property 有：'Focus'，'InitialState'，'InputDelay'，'SearchDirection'，'MaxIter'，'Tolerance'，'LimitError'，'FixedParameter' 和 'Trace'。

8．bj

功能：基于预测误差方法的 BJ 模型辨识。

语法：
```
m = bj(data,orders)
m = bj(data,'nb',nb,'nc',nc,'nd',nd,'nf',nf,'nk',nk)
m = bj(data,orders,'Property1',Value1,'Property2',Value2,...)
```

说明：bj 函数使用预测误差的方法估计 Box-Jenkins 模型：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$

data 是包括输出输入数据的基本 iddata 对象。

模型的阶次可以通过(...，'nb'，nb，'nc'，nc，'nd'，nd，'nf'，nf，'nk'，nk，...)定义，或利用参数 order 定义：orders = [nb nc nd nf nk]

nb，nc，nc 和 nf 为 Box-Jenkins 模型的阶次，nk 为延迟。特别有的：

$$nb:\quad B(q)=b_1+b_2q^{-1}+\cdots+b_{nb}q^{-nb+1}$$

$$nc:\quad C(q)=1+c_1q^{-1}+\cdots+c_{nc}q^{-nc}$$

$$nd:\quad D(q)=1+d_1q^{-1}+\cdots+d_{nd}q^{-nd}$$

$$nf:\quad F(q)=1+f_1q^{-1}+\cdots+f_{nf}q^{-nf}$$

【例 5】

```
B = [0 1 0.5];
C = [1 -1 0.2];
D = [1 1.5 0.7];
F = [1 -1.5 0.7];
m0 = idpoly(1,B,C,D,F,0.1);
e = iddata([],randn(200,1));
u = iddata([],idinput(200));
y = sim(m0,[u e]);
z = [y u];
mi = bj(z,[2 2 2 2 1],'MaxIter',0)
m = bj(z,mi)
m.EstimationInfo
m = bj(z,m);
compare(z,m,mi)
```

9．n4sid

功能：基于子空间方法的状态空间模型辨识。

语法：m = n4sid(data)

m = n4sid(data,order,'Property1',Value1,...,'PropertyN',ValueN)

说明：n4sid 用于估计状态空间形式，并返回一个 idss 对象。它可以处理任意多的输入输出，包括时间序列（没有输入）。状态空间模型具有如下形式：

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t)$$

$$y(t) = Cx(t) + Du(t) + e(t)$$

m 为返回的 idss 对象；

data 为包含输入输出数据的 iddata 对象；

order 为状态空间模型所需的阶次，为行向量的形式，缺省为 order=[1:10],即模型的阶次为 1～10。

函数 n4sid 对所有指定的阶次进行状态空间模型辨识，并绘图比较不同阶次模型的脉冲响应的 Hankel 矩阵奇异值，让用户自己选择模型的阶次。

【例 6】 建立一个三输入二输出的对象模型，尝试不同的阶次选择，并观察模型频率的响应。

```
z = iddata([y1 y2],[ u1 u2 u3]);
m = n4sid(z,5,'n4h',[7:15]','trace','on');
bode(m,'sd',3)
```

10. pem

功能：采用预测误差算法辨识一般线性输入输出模型。

语法：m = pem(data)

m = pem(data,mi)

m = pem(data,mi,'Property1',Value1,...,'PropertyN',ValueN)

m = pem(data,orders)

m = pem(data,'nx',ssorder)

m = pem(data,'na',na,'nb',nb,'nc',nc,'nd',nd,'nf',nf,'nk',nk)

m = pem(data,orders,'Property1',Value1,...,'PropertyN',ValueN)

说明：pem 是工具箱中一个基本的估计函数，它包含了许多情况。输入参数定义：

data：包含输入输出数据的 iddata 对象。

具有初始化条件的模型：

mi 为任何 idmodel 对象，idarx，idpoly，idss 或 idgrey。

m 为由 mi 定义的模型结构中最适合的模型。

黑箱状态空间模型：

m=pem（data， n），n 为正整数。

orders：用于指定线性输入输出模型中各个多项式的阶次和纯时延大小，order=[na nb nc nd nf nk]。

函数 pem 可以处理一般的多输入-单输出系统模型：

$$A(q)y(t) = \frac{B_1(q)}{F_1(q)}u_1(t-nk_1) + \cdots + \frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu}) + \frac{C(q)}{D(q)}e(t)$$

【例 7】 考虑一个三输入二输出的对象模型。

```
z = iddata([y1 y2],[ u1 u2 u3]);
m = pem(z,5,'ss','can')
```

2.2.4 递推参数模型辨识函数介绍

由于基于一次完成算法的参数模型辨识函数内存占用量很大、难于用于在线辨识的缺点，因此在 MATLAB 中还提供了基于递推公式的参数模型辨识，见表 2-5。

表 2-5 递推参数模型辨识函数

函 数	功 能 说 明
rarx	基于递推最小二乘法的 ARX 模型辨识
rarmax	采用递推算法进行 ARMAX 模型辨识
rbj	Box-Jenkins 模型的递推辨识
roe	输入误差模型的递推辨识
rpem	基于递推预测方法的线性输入输出模型辨识
rplr	基于伪线性回归的一般线性输入输出模型辨识
segment	基于数据分段的 ARX、ARMAX 模型辨识

1．rarx

功能：基于递推最小二乘法的 ARX 模型辨识。

语法：thm = rarx(z,nn,adm,adg)

[thm,yhat,P,phi] = rarx(z,nn,adm,adg,th0,P0,phi0)

说明：输入参数的定义：

z：对象输入输出数据向量，z=[y u]。其中 y 为对象的输出数据列向量，u 为对象的输入数据列向量，当 z=y 时，函数对 AR 模型进行辨识；

nn：指定 AR 或 ARX 模型的阶次。AR 辨识时为 *nn=na*；ARX 模型辨识时 *nn=[na nb nk]*。其中 *na*，*nb* 为多项式 $A(q)$和 $B(q)$的阶次，*nk* 为对象的纯时延；

adm,adg：用于指定递推最小二乘法的类型；

- adm='ff',adg='lam'
- adm='ug',adg=gam
- adm='ng',adg=gam
- adm='kf',adg=R1

th0：指定模型参数的初始值，为行向量；

P0：指定参数估计的协方差矩阵初值；

Phi0：数据向量的初始值，缺省为 0 向量，定义为：

$\varphi(t) = [\ y\ (\ t-1)\ ,\ldots,\ y\ (\ t-na\ )\ ,\ u\ (\ t-1),\ldots,\ u\ (t-nb-bk+1)]$

输出参数的定义：

thm：参数估值矩阵 thm(k,:) = [a1,a2,...,ana,b1,...,bnb,c1,...,cnc]；

yhat：输出的当前预测值矩阵；

P：当前参数估计的协方差矩阵；

phi：当前的数据向量；

psi：梯度向量。

【例 1】 采用 AR 辨识模型消除噪声，设测量到的信号为：

y(t)=2*sin(t)-1.5*sin(t-1)+e(t)

使用如下程序消除噪声

```
r=sin(0.1:0.1:31)
r1=r(1:300)
r2=r(2:301)
e = randn(1,300)
y=1.*r1-1.5.*r2+0.1.*e
z = [y'r1'];
[thm,noise] = rarx(z,[0 6 1],'ng',0.1);
plot(noise)
```

信号图形如图 2-11 所示。

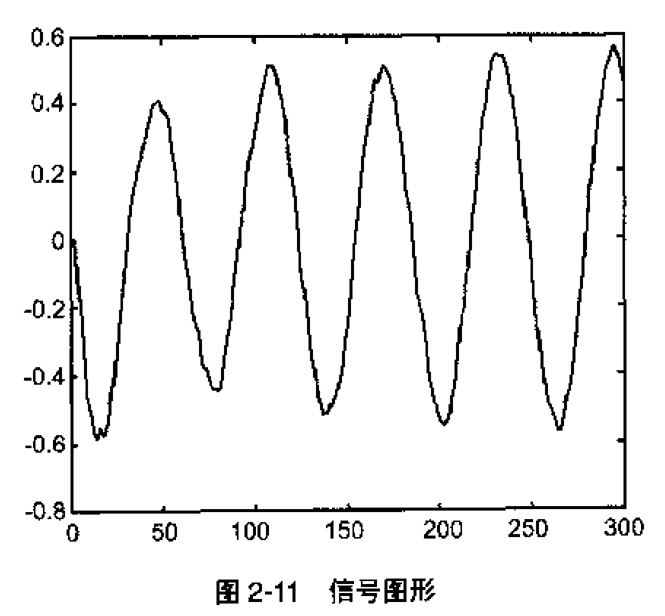


图 2-11 信号图形

2. rarmax

功能：采用递推算法进行 ARMAX 模型辨识。

语法：thm = rarmax(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rarmax(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：该函数只能用于单输入输出对象的辨识。

输入参数定义为：

z：对象的输入输出数据向量，z=[y u]。其中 y 为对象的是输出数据列向量，u 为对象的输入数据列向量。z=y 时为 AR 模型辨识；

nn：指定 ARX 或 AR 模型的阶次。当为 ARX 模型时 nn=[na nb nk]。其中 na，nb 为 ARX 模型中多项式 A(q) 和 C(q)的阶次，nk 为对象的纯时延；当为 AR 模型时，nn=na。特别的：

$$na: \quad A(q)=1+a_1q^{-1}+\cdots+a_{na}q^{-na}$$

$$nb: \quad B(q)=b_1+b_2q^{-1}+\cdots+b_{nb}q^{-nb+1}$$

adm,adg,th0,P0,phi0,psi0 的定义同函数 rarmax。

输出参数定义：

thm：参数估值矩阵 thm(k,:) = [a1,a2,...,ana,b1,...,bnb]；

yhat：输出的当前预测值矩阵；

P：当前参数估计的协方差矩阵；

phi：当前的数据向量；

psi：梯度向量。

【例 2】 我们考虑对测量信号进行消除噪声处理问题。假设信号如下：

y(t)=2*sin(t)−1.5*sin(t−1)+0.2*e(t)−0.1*e(t−1)

其中 e(t)为平均值为 0 的噪声，用 rarmax 消除噪声的程序。结果如图 2-12 所示。

```
r=sin(0.1:0.1:31);
a=r(1:300);
b=r(2:301);
e = randn(1,301);
c = e(1:300);
d= e(2:301);
y=2.*a-1.5.*b+0.2.*c-0.1.*d;
z = [y' a'];
[thm,yp] = rarmax(z,[0 6 2 1],'ng',0.1);
plot(yp)
```

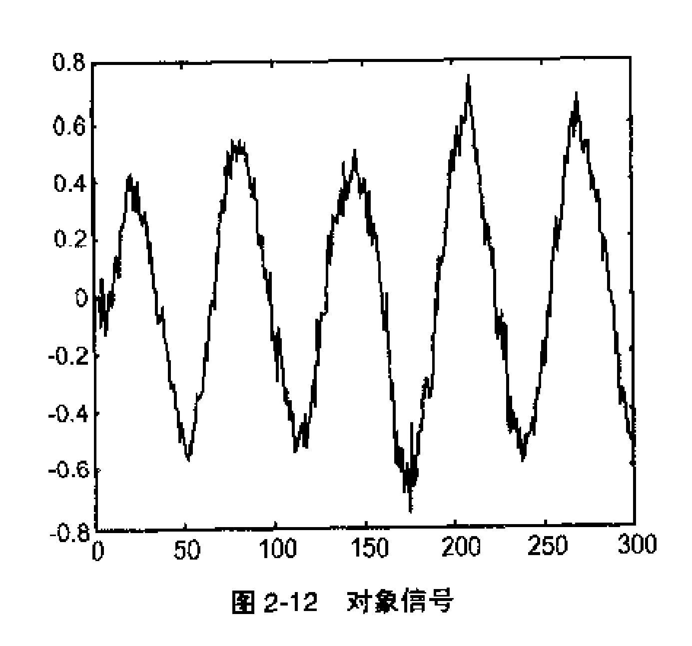


图 2-12 对象信号

3. rbj

功能：Box-Jenkins 模型的递推辨识。

语法：thm = rbj(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rbj(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：Box-Jenkins 模型的结构：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$

可以通过 rbj 来实现对该模型的递推辨识。

输入参数的定义：

z：对象输入输出数据向量，z=[y u]，其中 y 为输入，u 为输出，均为列向量的形式，z 也可为标准包含输入输出数据的 iddata 对象；

u：为对象输入数据列向量；

nn：指定 Box-Jenkins 模型的阶次，nn=[nb nc nd nf nk]，其中 nb、nc、nd 和 nf 分别为 Box-Jenkins 模型中 B(q)、C(q)、D(q)和 F(q)的阶次，nk 为对象的纯时延；

adm,adg, th0,P0,phi0,psi0 的定义同函数 rarx 的参数定义。

输出参数定义：

thm：参数估值矩阵

thm(k,:) = [b1,...,bnb,c1,...,cnc,d1,...,dnd,f1,...,fnf]；

yhat：输出的当前预测值矩阵；

P： 当前参数估计的协方差矩阵；

phi：当前的数据向量；

psi：梯度向量。

4. roe

功能：输入误差模型的递推辨识。

语法：thm = roe(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = roe(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：输入参数的定义：

z：对象输入输出数据向量，z=[y u]，其中 y 为输入，u 为输出，均为列向量的形式，z 也可为标准包含输入输出数据的 iddata 对象；

u：为对象输入数据列向量；

nn：指定 Box-Jenkins 模型的阶次，nn=[nb nf nk]，其中 nb、nc、nd 和 nf 分别为 Box-Jenkins 模型中 B(q)和 F(q)的阶次，nk 为对象的纯时延；

adm,adg, th0,P0,phi0,psi0 的定义同函数 rarx 的参数定义。

输出参数定义：

thm：参数估值矩阵 thm(k,:) = [b1,...,bnb,f1,...,fnf]；

yhat：输出的当前预测值矩阵；

P：当前参数估计的协方差矩阵；

phi：当前的数据向量；

psi：梯度向量。

5. rpem

功能：基于递推预测方法的线性输入输出模型辨识。

语法：thm = rpem(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rpem(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：输入参数说明：

参数 nn=[na nb nc nd nf nk],其中 na、nb、nc、nd 和 nf 为线性输入输出模型中对应的多项

式的次数，nk 为对象的纯时延。如果为多输入系统，则对应的阶次为行向量的形式。其他的深入输出参数同 rarx 和 rarmax。

输出参数说明：

thm：参数估值矩阵；

thm(k,:) = [a1,a2,...,ana,b1,...,bnb,...c1,...,cnc,d1,...,dnd,f1,...,fnf]；

yhat：输出的当前预测值矩阵；

P：当前参数估计的协方差矩阵；

phi：当前的数据向量；

psi：梯度向量。

6．rplr

功能：基于伪线性回归的一般线性输入输出模型辨识。

语法：thm = rplr(z,nn,adm,adg)

[thm,yhat,P,phi] = rplr(z,nn,adm,adg,th0,P0,phi0)

说明：函数的输入输出参数于 rpem 相同，但 rplr 只能用于单输入输出系统。

当应用于 ARMAX 模型辨识时：nn = [na nb nc 0 0 nk]；

当用于输出误差模型辨识时：nn = [0 nb 0 0 nf nk]。

7．segment

功能：基于数据分段的 ARX、ARMAX 模型辨识。

语法：segm = segment(z,nn)

[segm,V,thm,R2e] = segment(z,nn,R2,q,R1,M,th0,P0,ll,mu)

说明：输入参数定义：

z：对象的输入输出数据向量，z=[y u]。其中 y 为对象的是输出数据列向量，u 为对象的输入数据列向量。z=y 时为 AR 或 ARMA 模型辨识；

nn：用于指定模型的阶次和纯时延的大小，

- 对 ARMAX 模型 nn = [na nb nc nk]，
- 对 ARX 模型(nc=0) nn = [na nb nk]，
- 对 ARMA 模型 z = y ，nn = [na nc]，
- 对 AR 模型 nn = na；

R2：模型的息方差，缺省值为息方差的估值；

R1：模型参数发生突变时的参数协方差矩阵；

q：模型在任意时刻发挥上突变的概率，缺省值为 0.01；

M：指定同时采用的估计模型个数，缺省值为 5；

th0：指定参数的初始值，缺省值为 0；

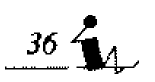
P0：初始的参数方差矩阵，缺省值为 10I；

ll：指定每个估计模型的最短有效时间，即在分段估计过程中的每个估计模型至少经过 ll 个时间步长才可能被其他模型代替，缺省值为 1；

mu：估计新息方差的遗忘因子；

输出参数定义为：

segm：基于分段估计的模型参数矩阵，该矩阵的第 k 行对应第 k 时刻具有最大后验概率



的参数估计；

V：分段参数模型的输出预测误差平方和；

thm：没有进行分段估计的模型参数矩阵；

R2e：新息方差的估值向量，第 k 个元素对应 k 时刻的估值。

【例 3】 下面给出函数 segment 对给定数据的分段结果，如图 2-13 所示。

```
y = sin([1:50]/3)';
thm = segment([y,ones(size(y))],[0 1 1],0.1);
plot([thm,y])
```

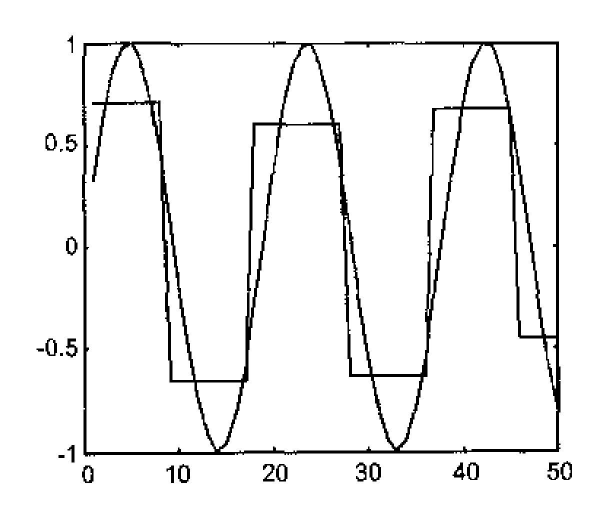


图 2-13 分段数据的信号

2.2.5 模型验证与仿真函数介绍

在建立模型和对模型进行辨识后，需要对结果进行检验，这就需要模型验证和仿真函数，MATLAB 系统辨识工具箱中提供的模型验证与仿真函数见表 2-6。

表 2-6 模型验证与仿真函数

函 数	功 能 说 明
idinput	生成信号，通常用作辨识的输入信号
sim	线性模型仿真
predict	根据历史数据预测 k 步后辨识模型的输出数据
compare	比较模型输出与实际输出
pe	计算模型预测误差
resid	计算模型预测误差并进行相关分析
aic	计算已辨识模型的 Akaike 信息标准
fpe	计算已辨识模型的 Akaike 最终预测误差

1. idinput

功能：生成信号，通常用作辨识的输入信号。

语法：u = idinput(N)

u = idinput(N,type,band,levels)

[u,freqs] = idinput(N, 'sine',band,levels,sinedata)

说明：输入参数的定义：

N：若 N 为数，则为生成信号数据的长度；若 N = [N nu]，则给出 nu 个输入信号，每个长度为 N；若 N = [P nu M]，则给出 nu 个周期输入信号，每个信号长度为 P*M，且具有周期 P。缺省为 nu=1 和 M =1。

type：生成信号的类型，包括如下几种：

- type='rgs'：高斯随机信号；
- type='rbs'：二值随机信号；
- type='prbs'：二值伪随机信号；
- type='sine'：和为正弦随机分布；

缺省值为'rbs'；

band：信号的带宽，当 type = 'rgs', 'rbs'和'sine'时该参数为行向量的形式 band = [wlow, whigh]，wlow 和 whigh 分别为信号频率带宽的下界和上界，缺省值为生成白噪声信号，即[0 1]；当 type='prbs'时，band = [0, B]，B 指定了信号在 1/B 长度的区间内为常数，缺省值同样为[0 1]；

levels：用于决定输入信号振幅的上下界，为行向量形式，即为：levels = [minu, maxu]，当 type='rgs'时，minu 为高斯信号的平均值减 1，maxu 为高斯信号平均值加一；

sinedata：用于输入信号类型为'sine'时的辅助变量，定义为：sinedata = [No_of_Sinusoids, No_of_Trials, Grid_Skip]其中 No_of_Sinusoids 用于生成信号的正弦函数的个数，No_of_Trials 用于决定在进行正弦函数相位随机化时直到获得信号幅值极小时的计算次数，缺省为 sinedata = [10,10,1]；Grid_Skip 用于控制奇次或偶次周期脉冲。

输出参数定义：

u：生成信号数据向量；

【例 1】 生成一个高斯随机信号。如图 2-14 所示。

```
u = idinput(200,'rgs',[0 1],[-1 1])
plot(u)
```

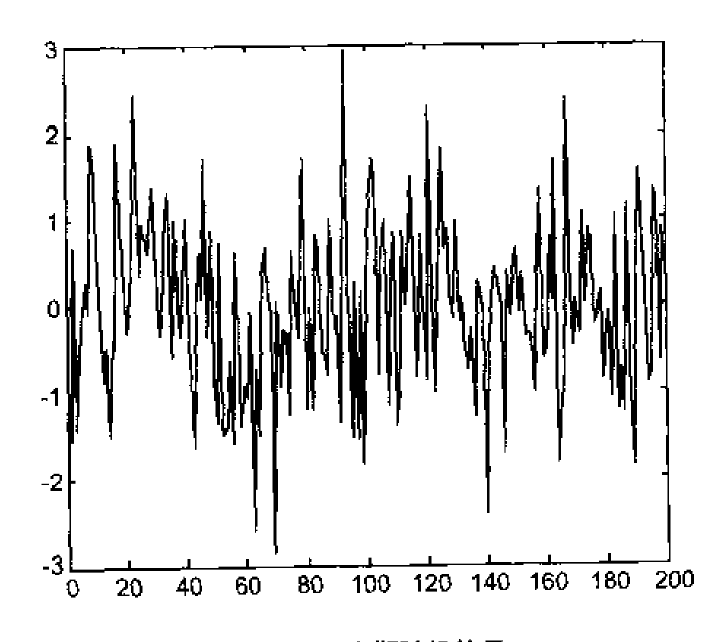


图 2-14 高斯随机信号

2. sim

功能：线性模型仿真。

语法：y = sim(m,ue)

[y, ysd] = sim(m,ue,init)

说明：输入参数定义：

m 为任何 idmodel 对象；

ue 为只包含输入的 iddata 对象，其中包含的输入信号的个数必须与 m 中的输入信号的个数相同；

init 给出初始化条件：

- init='m' (缺省值)使用模型 m 中原始的初始化条件；
- init = 'z' 使用零初始化条件；
- init = x0，其中 x0 为一列向量，并使用该向量作为初始值。

输出参数定义：

函数返回值 y 为仿真输出，为一 iddata 对象；

ysd：为仿真输出的标准差。

【例 2】 如图 2-15 所示。

```
A = [1 -1.5 0.7]; B = [0 1 0.5];
m0 = idpoly(A,B);
e = iddata([],randn(500,1));
u = iddata([],idinput(500,'prbs'));
y = sim(m0,[u e]);
z = [y u];
plot(y)
```

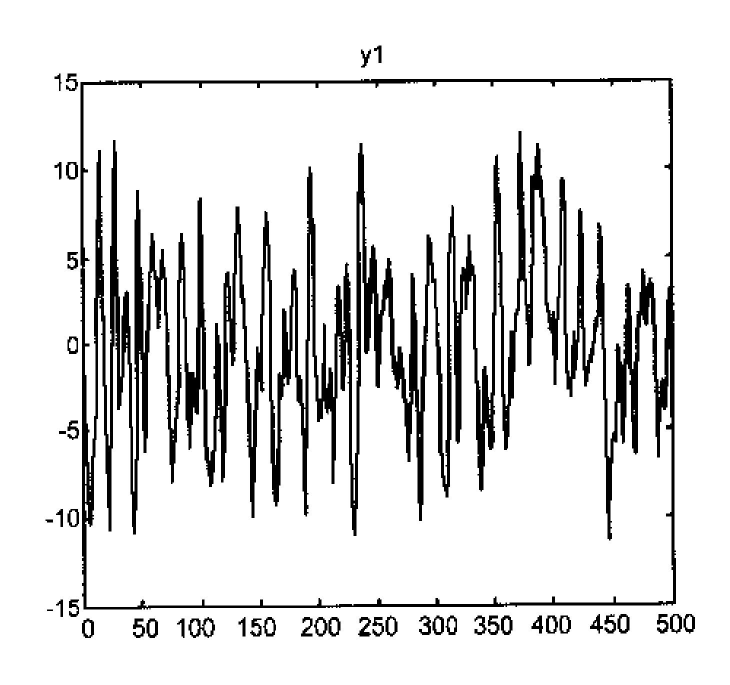


图 2-15　线性模型仿真

3. predict

功能：根据历史数据预测 k 步后辨识模型的输出数据。

语法：yp = predict(m,data)

[yp,mpred] = predict(m,data,k,init)

说明：输入参数定义：

m：任何 idmodel 对象(idpoly，idss，idgrey 或 idarx)；

data：包含输入输出数据的 iddata 对象；

k：预测时间长度；

init：决定如何处理初始状态

- init ='estimate'：取初始化状态的值，使得模型的预测误差范数最小；
- init ='zero'：初始化状态取零；
- init ='model'：使用模型内部的初始状态；
- init = x0：x0 为一列向量，并用作初始状态的值。

输出参数定义：

yp：模型的预测输出，为包含预测值的 iddata 对象；

mpred：k 步预测计算的模型。

【例 3】 如图 2-16 和图 2-17 所示。

```
m0 = idpoly([1 -0.99],[],[1 -1 0.2]);
e = iddata([],randn(400,1));
y = sim(m0,e);
m = armax(y(1:200),[1 2]);
yp = predict(m,y,4);
plot(y(201:400))
plot(yp(201:400))
```

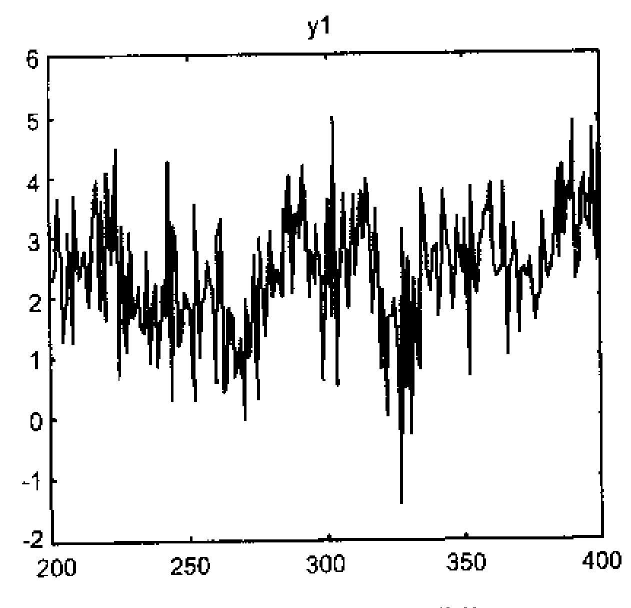


图 2-16 对象实际输出曲线

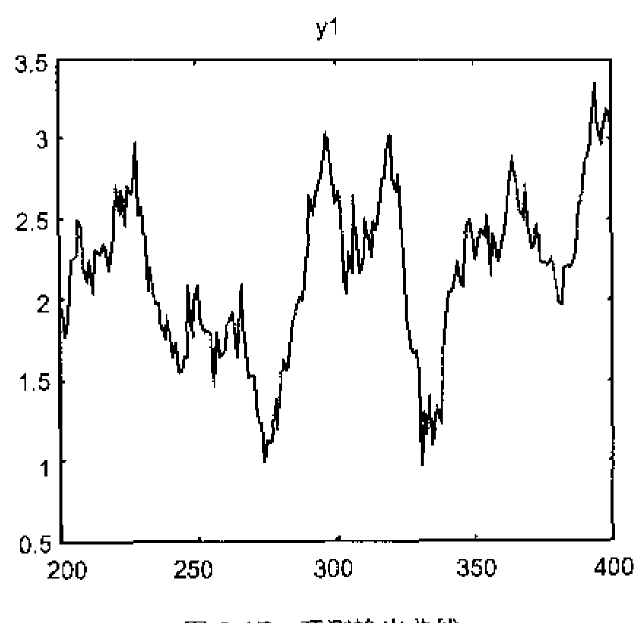


图 2-17　预测输出曲线

4．compare

功能：比较模型输出与实际输出。

语法：compare(data,m);

compare(data,m,k,sampnr,init)

compare(data,m1,m2,...,mN,Yplots)

compare(data,m1,'PlotStyle1',...,mN,'PlotStyleN',k,sampnr,init)

[yh,fit] = compare(data,m1,'PlotStyle1',...,mN,'PlotStyleN',k,sampnr,init)

说明：输入参数说明：

data：包含输入输出参数数据的 iddata 对象；

m：仿真模型对象；

k：预测的时间长度；

sampnr：指出用于计算模型适应度和绘图所用的数据点向量；

Yplots：为一个字符串类型的数组，只有输出名出现在该数组中时，才作图。假如该参数没有指定，则作出所有图；

init：指明如何处理模型的初始条件：

init = 'e' ('estimate')估计最合适的初始条件；

init = 'm' ('model') 使用模型内部的初始条件；

init = 'z' ('zero') 使用零初始条件；

init = x0，x0 为和模型状态向量同样长度的列向量，并用 x0 作为初始条件；

init = 'e'为缺省值。

输出参数说明：

yh：模型输出的 iddata 对象数组，每个元素包含对应模型的输出数据；

fit：在一般情况下为一个三维数组，fit(kexp,kmod,ky)，包含实验 kexp，模型 kmod 和输出 ky 的适应度值。

【例 4】

```
ze = z(1:250);
zv = z(251:500);
m= armax(ze,[2 3 1 0]);
compare(zv,m,6);
```

5. pe

功能：计算模型预测误差。

语法：e = pe(m,data)

[e,x0] = pe(m,data,init)

说明：输入参数定义：

data：包含输入输出数据的 iddata 对象；

m：任何 idmodel 模型对象；

init：决定如何处理初始条件：

init='estimate'：取该初值可是预测误差范数最小，并把该初始值返回在 x0 中；

init='zero'：取零初值；

init='model'：取模型内部储存的初值；

init=x0：x0 为和初值相同维数的列向量，并用 x0 作为初始值。

输出参数定义：

e：iddata 对象，e.OutputData 包含了模型的预测误差。

6. resid

功能：计算模型预测误差并进行相关分析。

语法：resid(m,data)

resid(m,data,Type)

resid(m,data,Type,M)

e = resid(m,data);

说明：输入参数定义：

data：包含输入输出数据的 iddata 对象；

m：任何 idmodel 对象，通过给定数据被估计的对象。

Type 可以取如下字符串之一：

Type='Corr'：计算并显示 e 的自动修正函数以及 e 和输入 u 的交叉修正函数，为缺省值；

Type='ir'：作出脉冲响应图；

Type='fr'：作出频率响应的波特图。

7. aic

功能：计算已估计模型的 Akaike 信息理论指标。

语法：am = aic(Model)

说明：Model 为任意已估计的 idmodel 对象（idarx, idgrey, idpoly, idss）。

返回值 am 为 Akaike 信息理论指标：

$$AIC = \log(V) + 2d/N$$

V 为损失函数，d 为估计参数的个数，N 为估计数据的个数。

8．fpe

功能：计算已估计模型的 Akaike 最终预测误差。

语法：am = fpe(Model)

说明：Model 为任意已估计的 idmodel 对象（idarx, idgrey, idpoly, idss）。

返回值 am 为 Akaike 最终预测误差：

$$FPE = V(1 + d/N) / (1 + d/N)$$

V 为损失函数，d 为估计参数的个数，N 为估计数据的个数。

2.2.6　其他常用函数介绍

在系统辨识工具箱中包含如下的作图函数：

1．plot

功能：绘制输入输出 iddata 对象。

语法：
```
plot(data)
plot(d1,...,dN)
plot(d1,PlotStyle1,...,dN,PlotStyleN)
```

说明：data 为包含输入输出数据的 iddata 对象。d1,…,dN，为一列具有相同输入输出名的 iddata 对象；

PlotStyleN：指定绘图所用的颜色、线条等。

2．bode

功能：绘制波特图。

语法：
```
bode(m)
[mag,phase,w] = bode(m)
[mag,phase,w,sdmag,sdphase] = bode(m)
bode(m1,m2,m3,...,w)
bode(m1,'PlotStyle1',m2,'PlotStyle2',...)
bode(m1,m2,m3,..'sd',sd,'mode',mode,'ap',ap)
```

说明：m,m1,m2,m3,…：任意 idmodel 对象；

参数 sd，ap，mode 和 w 可以以任意顺序出现，其意义如下：

sd：如果 sd>0 函数的置信区间在图中用点线画出；

ap：通常振幅和相位同时绘制在图中。'ap'='A'只绘制振幅曲线，'ap'='P'则只绘制相位曲线，'ap'='B'两者都画；

PlotStyleN：指定绘图所用的颜色、线条等；

mode='same'：所有图将在一幅图中绘制出。

【例 1】　如图 2-18 所示。

```
a = [1 -0.5 0.7];
b = [0 1 0.5];
m0=idpoly(a,b);
```

```
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0,[u,e]);
z=[y,u];
m2 = iv4(z,[2 2 1]);
bode(m2,5,'B','same')
```

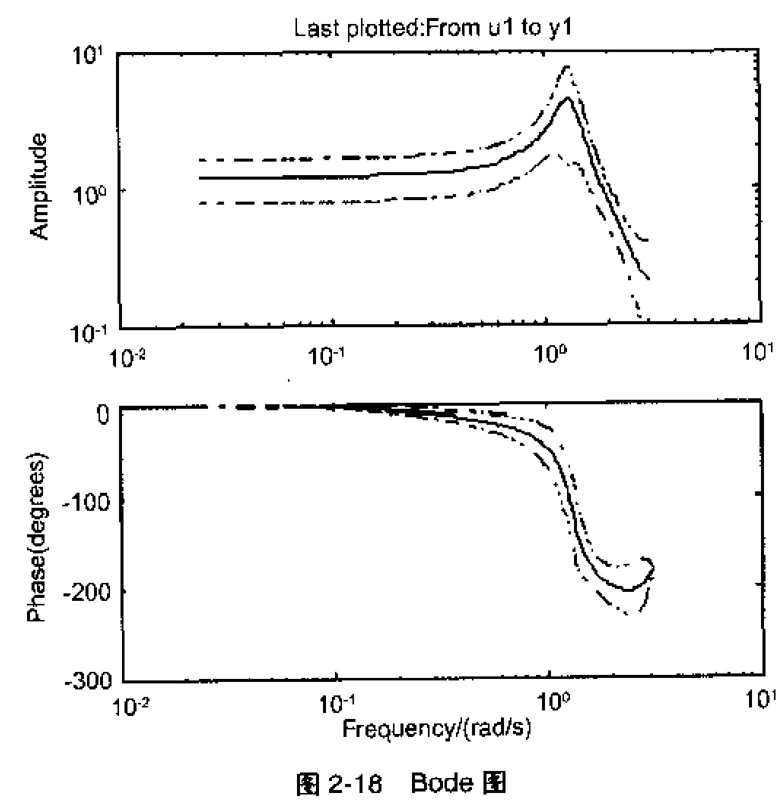


图 2-18　Bode 图

3. ffplot

功能：绘制频率响应曲线。

语法：ffplot(m)

```
[mag,phase,w] = ffplot(m)
[mag,phase,w,sdmag,sdphase] = ffplot(m)
ffplot(m1,m2,m3,...,w)
ffplot(m1,'PlotStyle1',m2,'PlotStyle2',...)
ffplot(m1,m2,m3,..'sd',sd,'mode',mode,'ap',ap)
```

说明：输入参数同 bode。

4. present

功能：显示模型的信息。

语法：present(m)

说明：m 为模型对象。函数将返回多项式的参数及其标准差、损失函数、AIC 指标等。

【例 2】

```
a = [1 -0.5 0.7];
```




```
b = [0 1 0.5];
m0=idpoly(a,b);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0,[u,e]);
z=[y,u];
m1 = arx(z,[2 2 1]);
m2 = iv4(z,[2 2 1]);
present(m1)
present(m2)
```

程序运行后显示关于模型 m1 的信息为：

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.4866 (+-0.0315) q^-1 + 0.687 (+-0.02763) q^-2
B(q) = 1.047 (+-0.06094) q^-1 + 0.3941 (+-0.07039) q^-2
Estimated using ARX from data set z
Loss function 1.0729 and FPE 1.10189
Sampling interval: 1
Created:        19-Sep-2002 13:15:20
Last modified: 19-Sep-2002 13:15:20
```

显示关于 m2 的信息为：

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.4562 (+-0.04523) q^-1 + 0.6827 (+-0.03665) q^-2
B(q) = 1.045 (+-0.06056) q^-1 + 0.4278 (+-0.07863) q^-2
Estimated using IV4 from data set z
Loss function 1.08456 and FPE 1.11387
Sampling interval: 1
Created:        19-Sep-2002 13:15:20
Last modified: 19-Sep-2002 13:15:20
```

5. nyquist

功能：绘制模型的 nyquist 曲线。

语法：

```
nyquist(m)
[fr,w] = nyquist(m)
[fr,w,covfr] = nyquist(m)
nyquist(m1,m2,m3,...,w)
nyquist(m1,'PlotStyle1',m2,'PlotStyle2',...)
nyquist(m1,m2,m3,..'sd*5',sd,'mode',mode)
```

说明：m 为任意 idmodel 对象；

mode='same'：所有图将在一幅图中绘制出；

w={wmin,wmax}用于指定作出该特定频率范围的图；

sd：如果 sd>0 函数的置信区间在图中用点线画出；

当 nyquist 被一个系统调用并返回输出参数时，将不作图；

如：fr = nyquist(m,w) or [fr,w,covfr] = nyquist(m)，假如 m 有 ny 个输出和 nu 个输入，w 包含 nw 个频率。则 fr 为 ny*nu*nw 维的矩阵。且 fr(ky,ku,k)给出从第 ku 个输入到第 ky 个输出在频率 w(k)出的复值频率响应。covfr 则为 2*2 的协方差矩阵，它的 1,1 部分为 fr 实部的方差，2,2 部分为虚部之间的方差，1,2 和 2,1 部分为虚部与实部之间的方差。

【例 3】 绘制对象的 nyquist 曲线，结果如图 2-19 所示。

```
a = [1 -0.5 0.7];
b = [0 1 0.5];
th0=idpoly(a,b);
nyquist(th0)
```

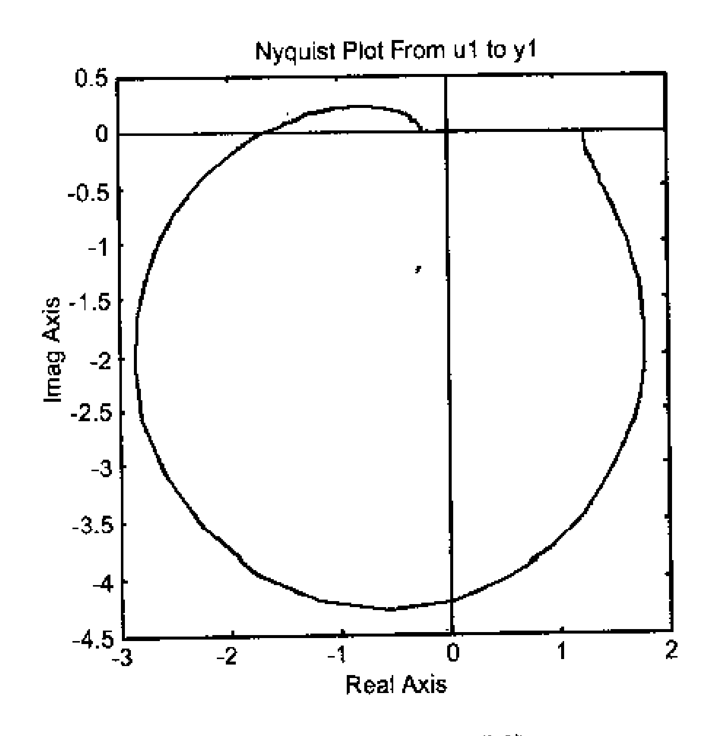


图 2-19 对象 nyquist 曲线

6. pzmap

功能：绘制零点和极点图。

语法：pzmap(m)

pzmap(m,'sd',sd)

pzmap(m1,m2,m3,...)

pzmap(m1,'PlotStyle1',m2,'PlotStyle2',...,'sd',sd)

pzmap(m1,m2,m3,..,'sd',sd,mode,axis)

说明：m 为任意 iddata 对象。

如果 sd>0 则零点和极点附近的置信区域也将被绘制出。

mode='sub'：为每个输入输出频道绘制一幅图；

mode='same'：所有的图将绘制在同一幅图中；

mode='sep'：绘制下一频道图时，擦掉前幅图。

缺省为'sub'。

axis = [x1 x2 y1 y2]，决定数轴的范围。

PlotStyleN：指定绘图所用的颜色、线条等。

【例 4】 结果如图 2-20 所示。

```
a = [1 -0.5 0.7];
b = [0 1 0.5; 0 1    0.7];
th0=idpoly(a,b);
pzmap(th0)
```

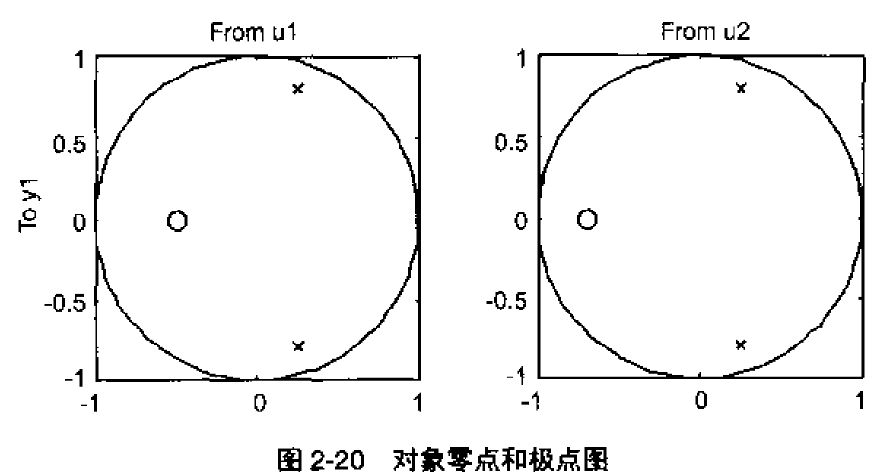


图 2-20　对象零点和极点图

7. impluse

功能：估计/计算/显示脉冲响应。

语法：impulse(m)

```
impulse(data)
impulse(data,'sd',sd,'pw',na,Time)
impulse(m,'sd',sd,Time)
impulse(m1,m2,...,dat1, ...,mN,Time,'sd',sd)
impulse(m1,'PlotStyle1',m2,'PlotStyle2',...,dat1,'PlotStylek',...,mN,'PlotStyleN',Time,'sd',sd)
[y,t,ysd] = impulse(m)
mod = impulse(data)
```

说明：

impluse 可以在任意 idmodel 和 iddata 对象上使用。

impulse(m)计算 m 的脉冲响应，并绘图，如果给出 sd 大于零，则绘制 0 点附近的置信区间。sd 对应于 standard deviations。

Time=[T1 T2]：T1 为开始时间，T2 为结束时间，如果不指定开始时间 T1，则缺省为-T2/4。

PlotStyle：为作图参数。

【例 5】 估计并对脉冲响应作图，结果如图 2-21 所示。

```
mc = idpoly(1,1,1,1,[1 1 0],'Ts',0);
impulse(mc)
```

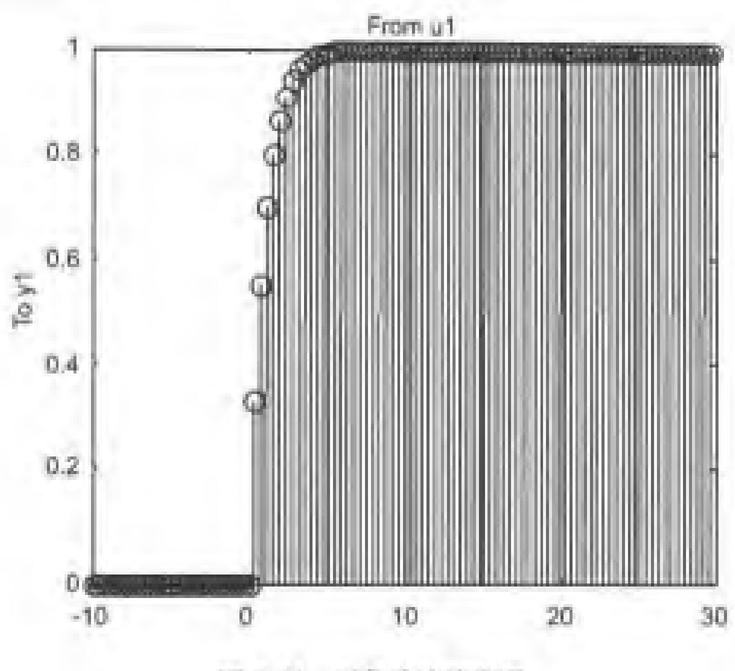


图 2-21　对象脉冲响应图

8. view

功能：绘制模型的特征（需要控制系统工具箱）。

语法：view(m)

```
view(m('n'))
view(m1,...,mN,Plottype)
view(m1,PlotStyle1,...,mN,PlotStyleN)
```

说明：m 为需要绘制的输入输出数据，为任何 idfrd 或 idmodel 模型。在经过适当的模型转换后，将打开控制系统工具箱的 LTI 观察器。

【例 6】

```
mc = idpoly(1,1,1,1,[1 1 0],'Ts',0);
view(mc)
```

打开的 LTI 观察窗口如图 2-22 所示。

可以在图上单击鼠标右键，可以选择绘制不同的图形，例如可以选择绘制 Nyquist 图，结果如图 2-23 所示。

系统辨识工具箱还提供了如下的模型选择函数：

1. arxstruc

功能：计算多个单输入 ARX 模型的损失函数。

语法：V = arxstruc(ze,zv,NN)

```
V = arxstruc(ze,zv,NN,maxsize)
```

说明：ze：用于模型辨识的输入输出数据向量或矩阵；

zv：用于模型验证的输入输出数据向量或矩阵；

NN：多个模型结构参数构成的矩阵，NN 每一行具有如下形式：nn = [na nb nk]；

maxsize：计算的辅助变量，参见 ar 函数；

V：V 的第一行为各个模型的损失函数值，其他为模型的结构参数。



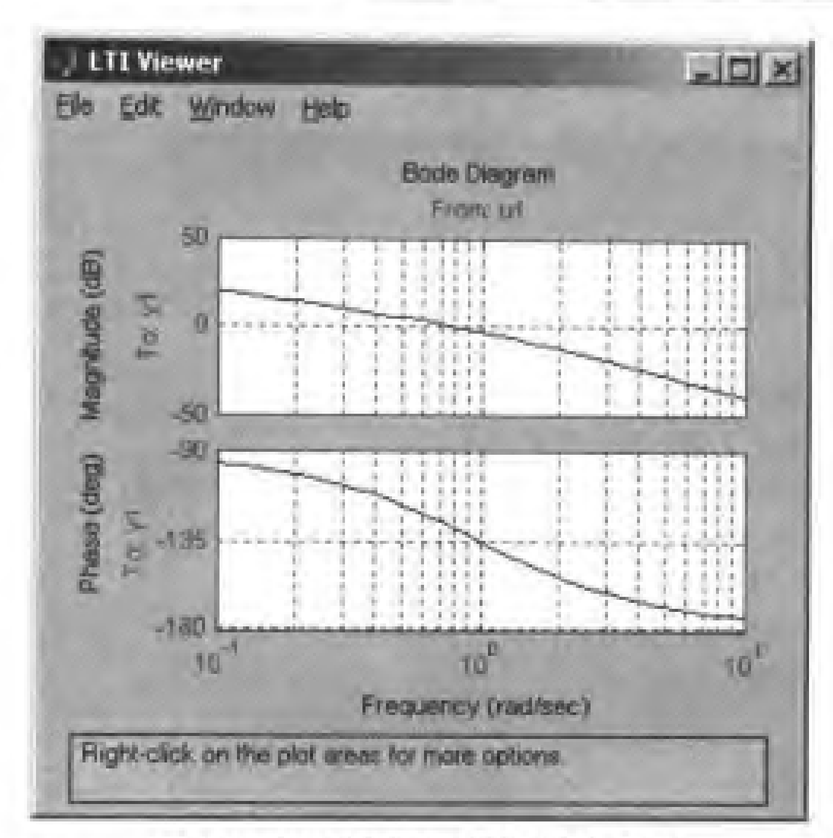


图 2-22 打开的 LTI 观察器绘制的 Bode 图

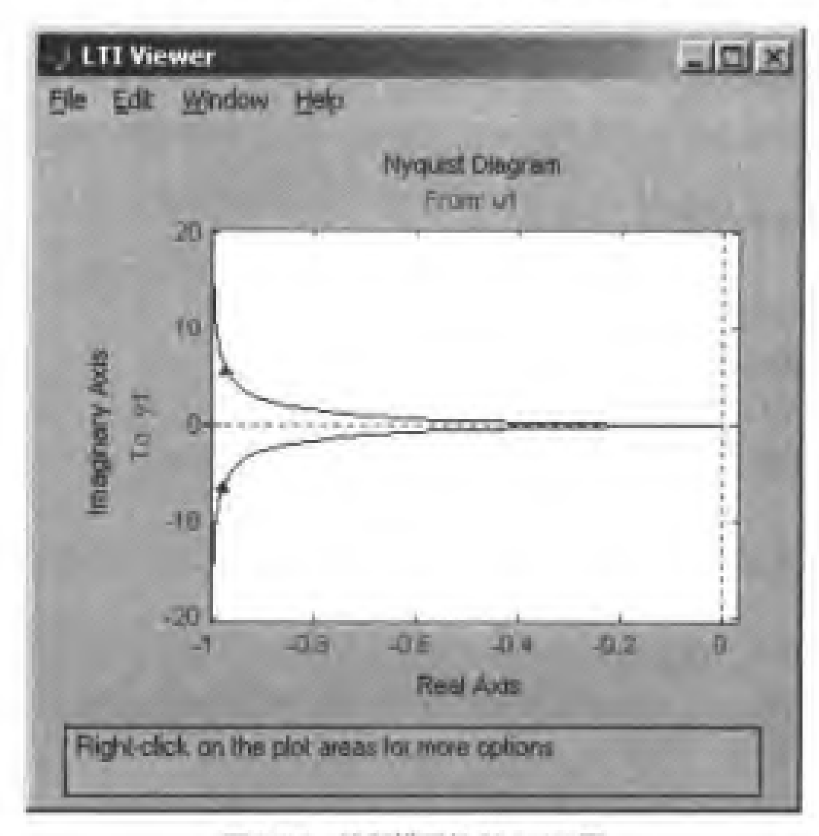


图 2-23 绘制模型的 Nyquist 图

【例 7】

NN = struc(1:5,1:5,1);

```
V   = arxstruc(z(1:200),z(201:400),NN);
```

2. ivstruc

功能：采用辅助变量方法计算多个 ARX 模型结构的损失函数。

语法：

```
v = ivstruc(ze,zv,NN)
v = ivstruc(ze,zv,NN,p,maxsize)
```

说明：

ze：用于模型辨识的输入输出数据向量或矩阵；

zv：用于模型验证的输入输出数据向量或矩阵；

NN：多个模型结构参数构成的矩阵，NN 每一行具有如下形式：

nn = [na nb nk];

maxsize：计算的辅助变量，参见 ar 函数。

p：决定是否计算矩阵的条件数，p=0 不计算；

v：各个模型的损失函数。

【例 8】

```
a = [1 -0.5 0.7];
b = [0 0 1];
th0=idpoly(a,b);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(th0,[u,e]);
z=[y,u];
v = ivstruc(z,z,struc(1:3,1:2,2:4))
```

3. selstruc

功能：模型结构选择

语法：nn = selstruc(v)

[nn,vmod] = selstruc(v,c)

说明：v：各个模型结构的损失函数，其格式见 arxstruc 函数；

c：可选参数，指定模型结构选择方式；

- c='plot'：为缺省值，绘制各个模型结构对应的损失函数值，由用户决定选择何种模型结构；
- c='log'：绘制各个模型结构对应的损失函数的对数值，由用户决定选择何种模型结构；
- c='aic'：按照极小化 Akaike 信息理论指标（AIC）选择模型结构：

$$V_{\text{mod}} = V\left(1 + \frac{2d}{N}\right)$$

其中 V 为损失函数，d 为模型结构中所有参数的总数，N 为估计所用的数据点的总数，该函数不作图；

- c='mdl'：按照极小化 Risanen 的最小描述长度指标选择模型结构：

$$V_{\text{mod}} = V\left(1 + \frac{d\log(N)}{N}\right)$$

参数意义同 AIC；

- c=n：n 为数量，则模型选择的指标为

$$V_{\text{mod}} = V\left(1 + \frac{cd}{N}\right)$$

参数意义同 AIC。

【例 9】 结果如图 2-24 所示。

```
a = [1 -0.5 0.7];
b = [0 1 0.5];
th0=idpoly(a,b);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(th0,[u,e]);
z=[y,u];
v1= arxstruc(z,z,struc(1:3,1:2,0:2));
v2= ivstruc(z,z,struc(1:3,1:2,0:2));
nn1 = selstruc(v1,'plot')
nn2=selstruc(v2,'plot')
```

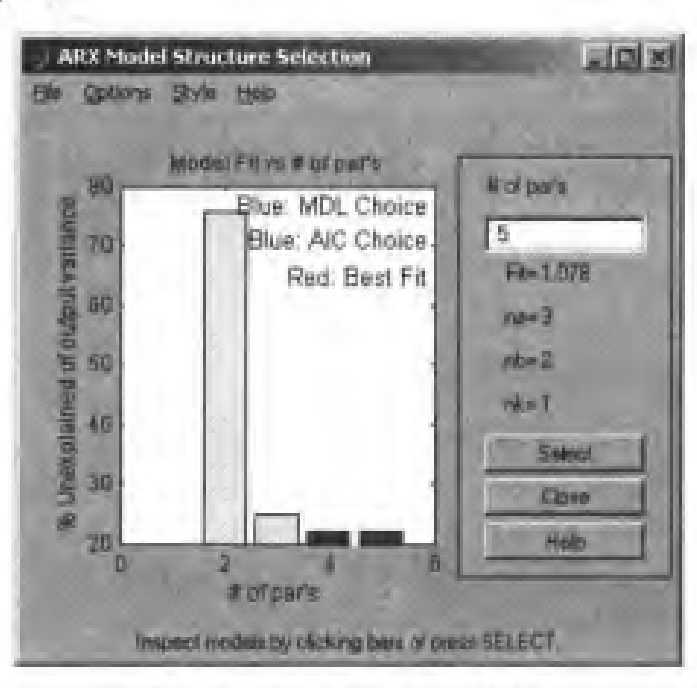


图 2-24　提供模型结构选择的窗口

4．struc

功能：生成多个模型结构参数。

语法：NN = struc(NA,NB,NK)

说明：NA：ARX 模型多项式 A(q)的阶次范围；

NB：ARX 模型多项式 B(q)的阶次范围；

NK：ARX 模型纯时延大小的范围；

NA，NB，NK 均为行向量；

NN：模型结构参数集矩阵。

【例 10】

```
NN = struc(1:2,1:2,4:5)
```

结果为：

```
NN =
      1     1     4
      1     1     5
      1     2     4
      1     2     5
      2     1     4
      2     1     5
      2     2     4
      2     2     5
```

系统辨识工具箱中包含的数据预处理函数有：

1. dtrend

功能：消除数据中的趋势项。

语法：zd = detrend(z)

zd = detrend(z,o,brkp)

说明：z：输出输入测量数据；

o：趋势项的阶次，缺省为 0，即每一项减去平均值；若 o=1 则从数据中减去通过线性回归得到的线性趋势项；

brkp：指定分段线性回归的分段点；

返回值 zd 为消除趋势后的数据。

【例 11】 结果如图 2-25 和图 2-26 所示。

```
a = [1 -0.2 0.3];
b = [0 1 2];
m0=idpoly(a,b);
u = iddata([],idinput(300,'rbs',[0 1],[2 4]));
e = iddata([],randn(300,1));
y = sim(m0,[u,e]);
z=[y,u];
```

2. idfilt

功能：用一般的滤波器或 Butterworth 滤波器对输入输出数据进行滤波。

语法：zf = idfilt(z,filter)

zf = idfilt(z,ord,Wn)

zf = idfilt(z,ord,causality)

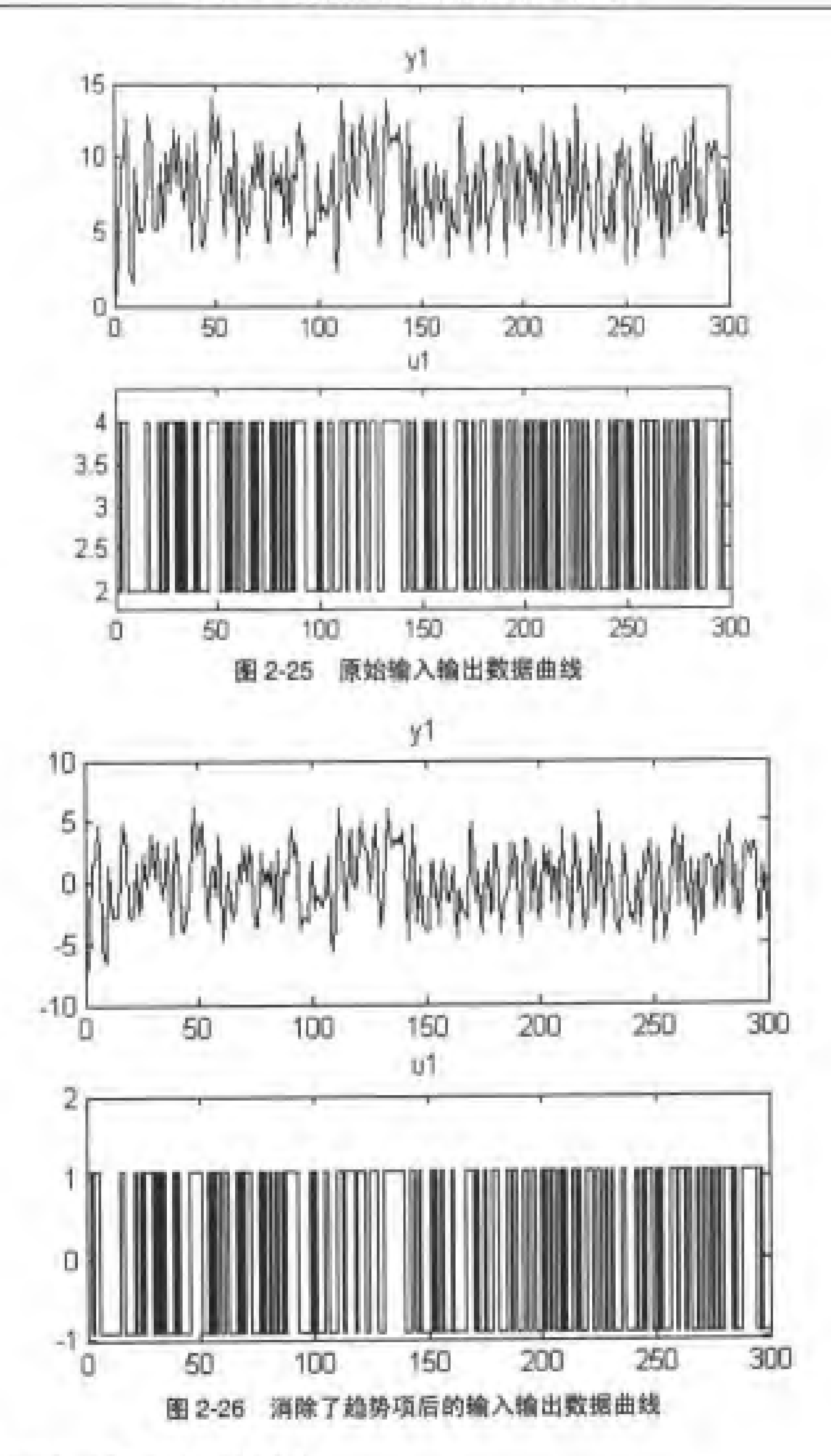


图 2-25　原始输入输出数据曲线

图 2-26　消除了趋势项后的输入输出数据曲线

[zf,mf] = idfilt(z,ord,Wn,hs)

说明：z：包含输入输出数据的 iddata 对象；

ord：指定 Butterworth 滤波器的阶次；

hs='high'时 Wn 用于指定高通滤波器的截止频率；

hs='stop'时 Wn=[Wl　Wh]用于指定带阻滤波器的频率范围；

hs 没指定时，若 Wn 只包含一个元素，则该参数用于指定低通滤波器的截止频率，若

Wn=[Wl Wh]则 Wn 低通滤波器的上下限频率。

输出参数定义：

zf：包含滤波后的输入输出数据的 iddata 对象；

mf：滤波器对应的 idmodel 对象。

【例 12】

```
a = [1 -0.2 0.3];
b = [0 1 2];
m0=idpoly(a,b);
u = iddata([],idinput(300,'rbs',[0 1],[2 4]));
e = iddata([],randn(300,1));
y = sim(m0,[u,e]);
z=[y,u];
zf=idfilt(z,2,4,'high');
plot(zf)
```

如图 2-27 所示。

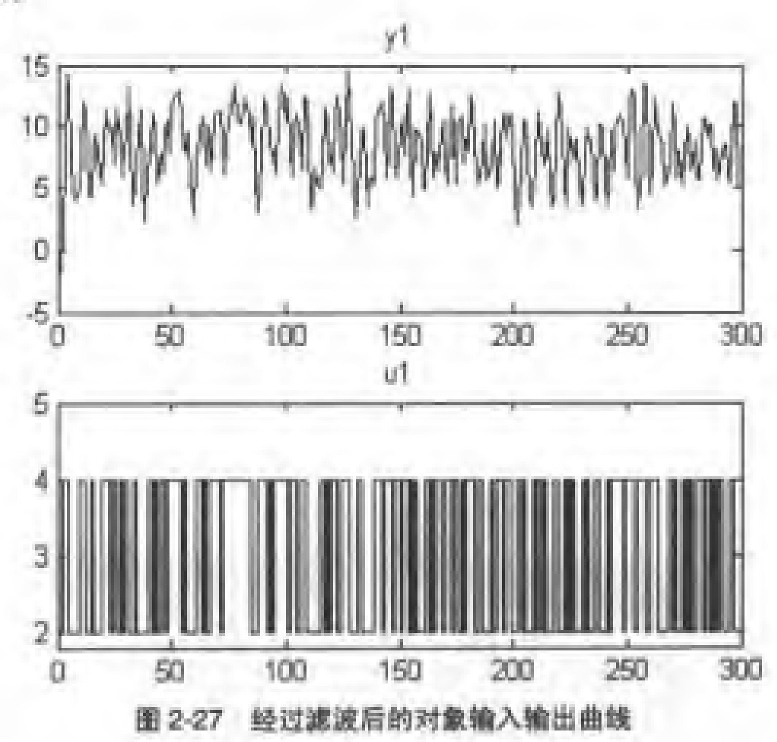


图 2-27　经过滤波后的对象输入输出曲线

3．resample

功能：对输入输出数据重新采样。

语法：datar = resample(data,P,Q)

datar = resample(data,P,Q,filter_order)

说明：

data：需要重新采样的 iddata 对象；

P 和 Q 为数，新的采样周期为原先采样周期的 Q/P 倍；

filter_order：指定在重新采样之前对数据进行滤波的滤波器的阶次，缺省值为 10；



输出参数 datar 为重新采样后的 iddata 对象。

4．simds

功能：具有不确定性的模型仿真

语法：simsd(m,u)

simsd(m,u,N,noise,Ky)

说明：m：任何 idmodel 对象；

u：包含输入的 iddata 对象；

N：仿真中随机生成的模型个数，这些随机模型按照 u 中有关参数的标准差的数据生成，缺省为 10；

noise：用于指定在仿真中是否加入噪声；

noise='noise'：加入噪声；

noise='nonoise'：不加入噪声。

【例 13】

```
a = [1 -0.2 0.7];
b = [0 1 0.5];
m0=idpoly(a,b);
u = iddata([],idinput(300,'rbs',[0 1],[2 4]));
e = iddata([],randn(300,1));
y = sim(m0,[u,e]);
z=[y,u];
i=iv4(z,[2 2 1]);
simsd(u,i,10,'nonoise')
```

绘制出的曲线如图 2-28 所示。

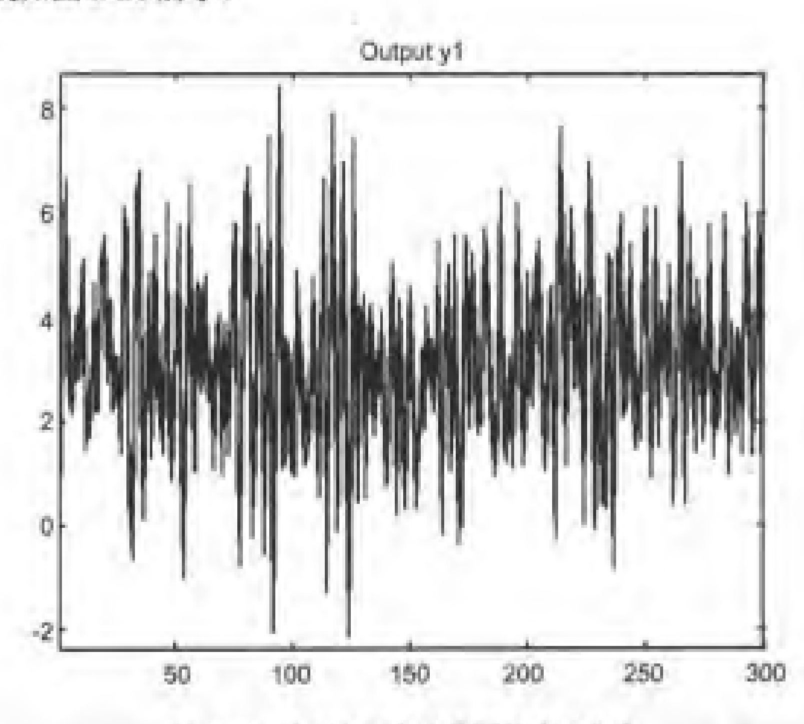


图 2-28　生成的不确定模型的输出曲线

2.3 系统辨识工具箱图形界面

MATLAB 除了在以命令和函数的方法提供许多系统辨识工具外，还提供了一个交互的图形界面。我们在 MATLAB 的命令窗口中键入 ident，即可进入系统辨识工具箱的图形界面。该工具箱的图形界面如图 2-29 所示。

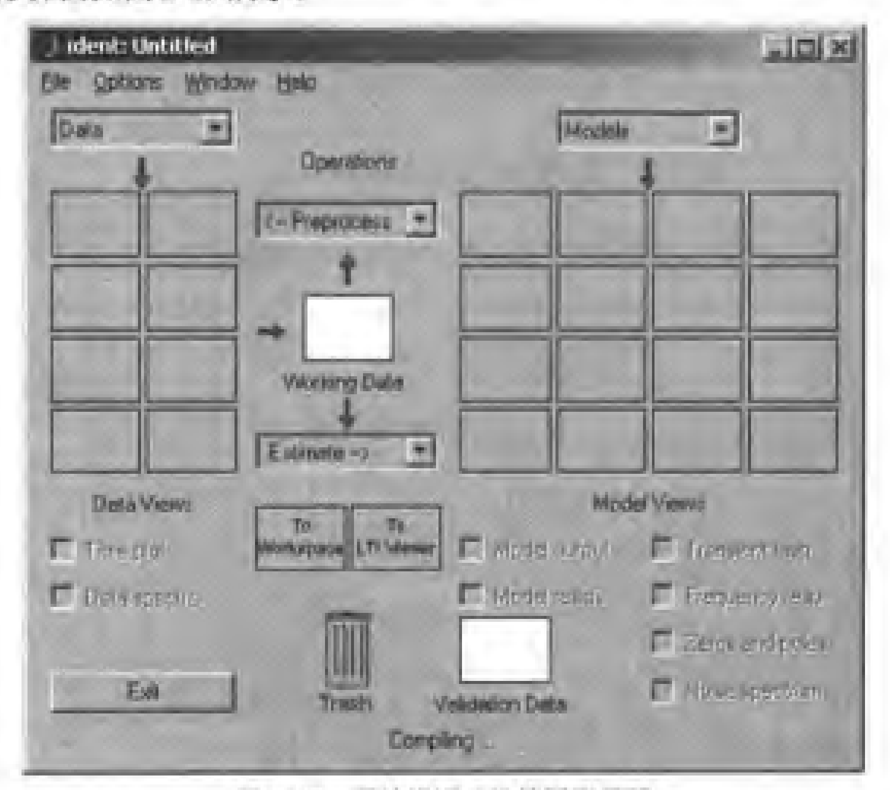


图 2-29 系统辨识工具箱图形界面

2.3.1 数据视图

在图中，左边为数据视图（Data Views）部分,在这一块中可以完成输入输出数据的导入，以及相关的绘图功能。选择左上角的下拉列表框中的 Import，就进入数据的导入界面，如图 2-30 所示。

在图的对话框中，通过指定 MATLAB 工作空间中对应的变量名称，即可导入输入输出数据。在导入数据以后，可以通过图形界面对这些数据进行处理。

下面举一个例子来说明系统辨识工具箱的功能和用法。

【例 1】 该例子的数据储存在 MATLAB 的 SID 文件："dryer.sid" 中（通常该文件位于：matlab\toolbox\ident\ident\目录的下面）。选择 File 菜单中的 Open Session 可以打开该文件。打开后系统自动导入数据，并进行预处理，此时界面如图 2-31 所示。

在图中，数据视图包括四组数据：Dryer,Dryerd,Dryerde 和 Dryerdv。其中 Dryer 为原始的输入输出数据，Dryerd 为消除了趋势项以后的输入输出数据，Dryerde 和 Dryerdv 分别用语模型边式和模型验证。可以通过鼠标单击一幅或几幅图表使得该图表成为视图区当前的图表，选种图表中的线条加粗显示。不用的图表可以拖动到中间下方的垃圾桶（Trash）内。

图 2-30　数据导入窗口

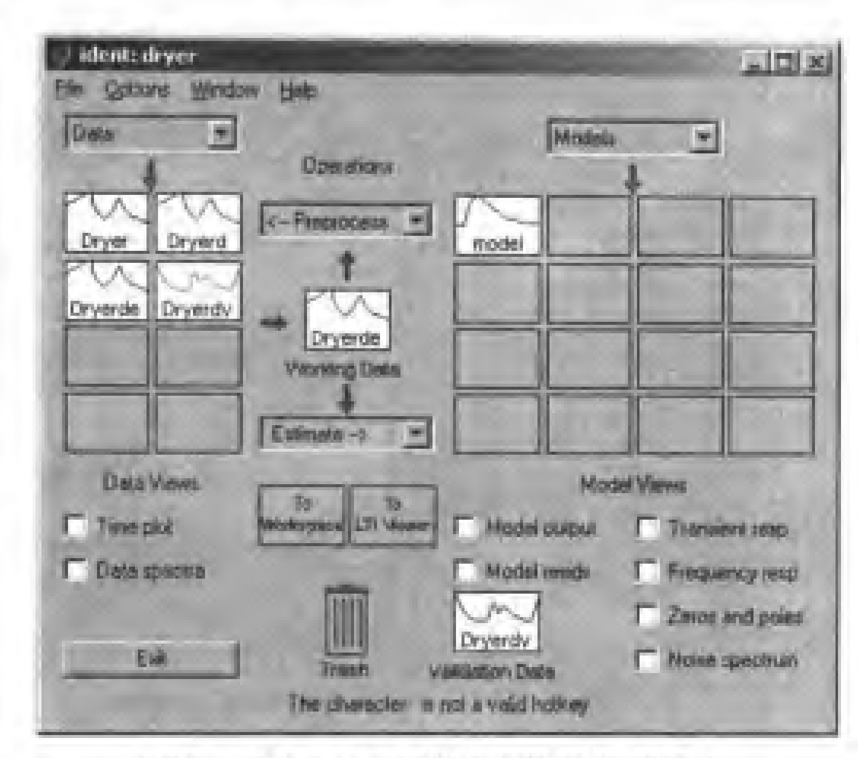


图 2-31　导入 dryer.sid 数据后的控制箱图形界面

同样也可以在 MATLAB 的工作空间中通过命令函数来完成这些功能，命令行为：

```
load dryer2
Dryerd = dtrend(Dryer,0)
Dryerde = Dryerd([1:500])
Dryerdv = Dryerd([501:1000])
```

函数 dtrend 用来消除输入输出数据中的趋势项，即所有值减去平均值。在图形界面的左下角有两个检查框，用于绘制数据时间变化曲线和频谱曲线，绘制出的曲线如图 2-32 和图 2-33 所示。

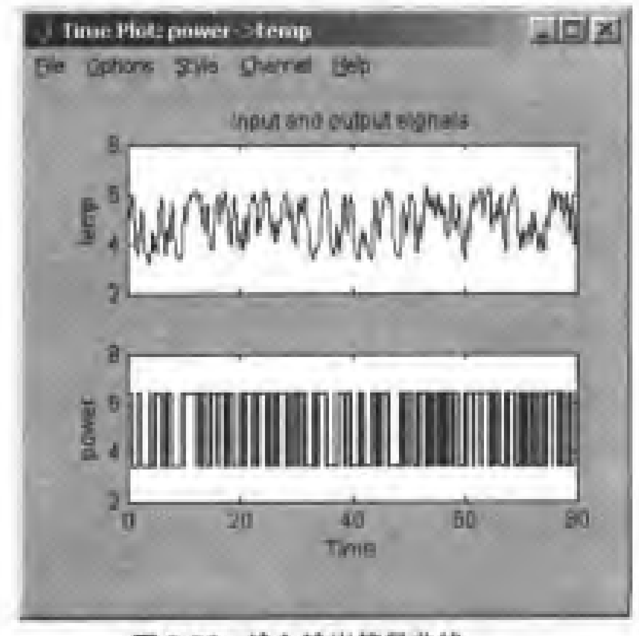


图 2-32　输入输出信号曲线

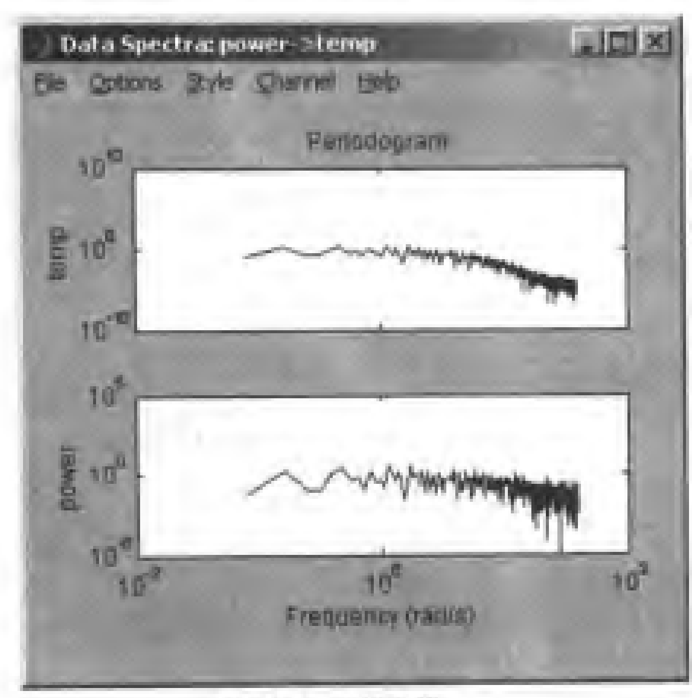


图 2-33 频谱曲线

2.3.2 操作选择

在图形界面的中间为数据操作部分，它包括两个下拉列表框。在上面的下拉列表框里可以选择对数据进行的操作，对输入输出数据进行有关的预处理，如滤波、消除趋势等。

下面的下拉列表框里可以选择模型的类型，并通过相应的对话框输入模型的阶次等信息。还可以将视图区的图表拖动至两个下拉列表框中间的的区域，使得该图表成为当前工作的数据。

例如：拖动 Dryerde 图表到当前工作数据区，并选择了消除趋势，绘制出的数据随时间变化的曲线如图 2-34 所示。

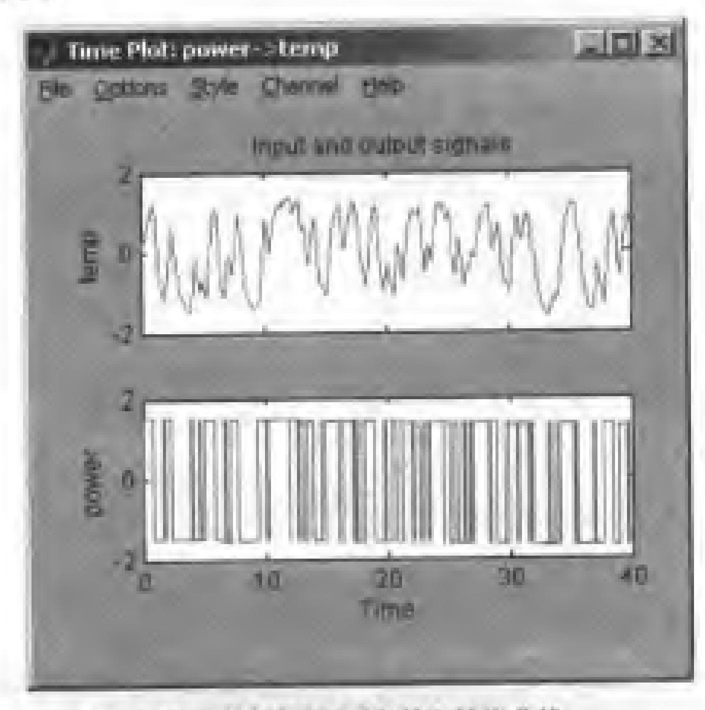


图 2-34 消除趋势后对象输入输出曲线

2.3.3 模型视图

在图形界面的右侧为模型视图区，这个区域的主要功能包括：选择不同的模型，并在不同模型间切换；进行模型的验证和特征曲线的绘制等。视图上方的下拉列表框中可以选择模型的种类。选定模型并输入相应的参数后，模型的图标就出现在下面的模型图表区域中，并可以通过鼠标单击来选择所需的模型种类。

模型视图的下方右许多检查框，其功能见表 2-7。

表 2-7 检查框功能

Model output	绘制模型输出曲线
Model resids	绘制模型预测残差曲线
Transient resp	绘制暂态响应曲线
Frequency resp	绘制频率响应曲线
Zeros and poles	绘制模型零点和极点图
Noise spectrum	绘制噪声频谱

例如：对应 dryer 的例子，采用 ARX 模型，绘制出模型输出曲线图如图 2-35 所示，频率响应曲线图如图 2-36 所示。

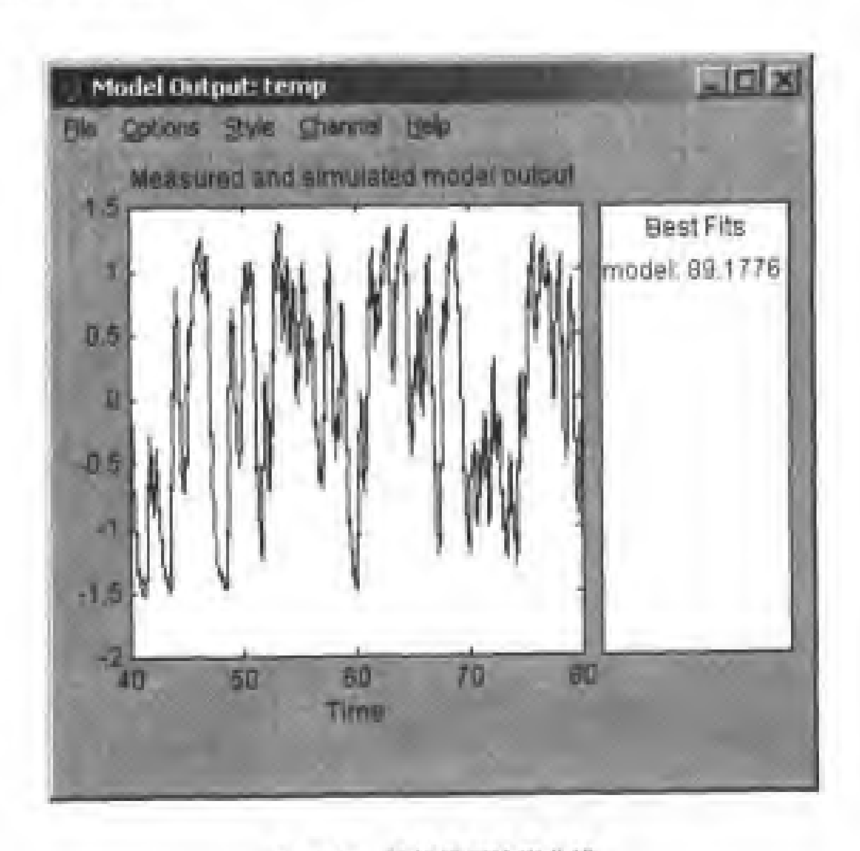


图 2-35 模拟模型输出曲线

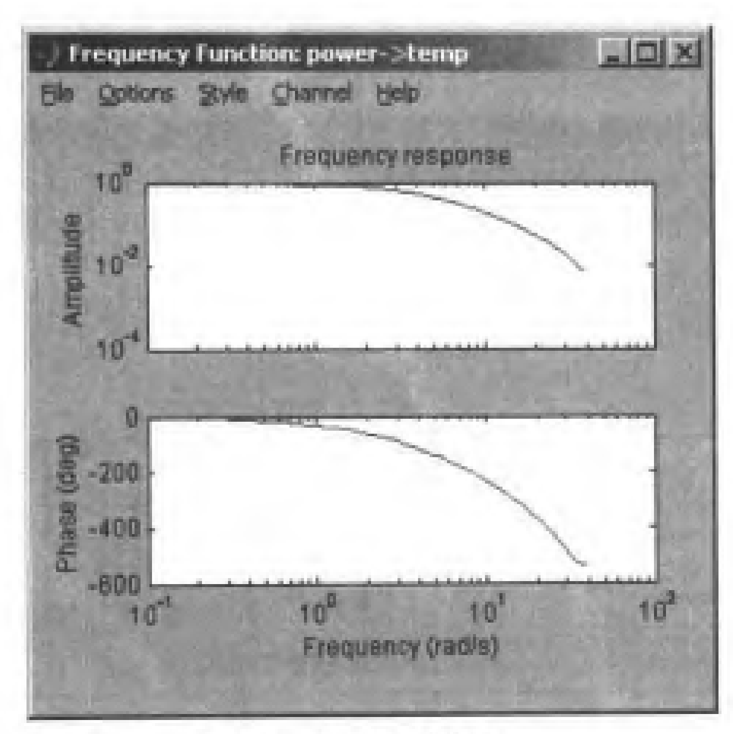


图 2-36　频率响应曲线图

第3章 控制系统工具箱

MATLAB 的控制系统工具箱主要处理以传递函数为主要特征的经典控制和以状态空间为主要特征的现代控制中的问题。该工具箱对控制系统，尤其是 LTI 线性时不变系统的建模、分析和设计提供了一个完整的解决方案。作为 MATLAB 最有利和最基本的工具箱之一，它的新版本不仅向后兼容，更重要的是其设计更合理，更易用。概括地说，控制系统工具箱具有以下几个方面的功能：

1. 模型的建立

通过控制系统工具箱提供的函数，可以很方便的建立传递函数、零极点增益、状态空间和频率响应模型，并可以实现任意两种模型之间的转换，并可以通过组合连接两种或多种系统从而实现一个复杂的系统模型。

2. 模型的分析

MATLAB 的控制系统工具箱支持对 MIMO 和 SISO 系统进行分析。系统提供了一些作图函数，使得对模型的分析更加直观，系统频率响应分析支持 Bode 图、Nichols 图、Nyquist 图。对时域响应，工具箱支持对系统的单位阶跃响应、单位脉冲响应、零输入响应以及更广泛的对任意信号进行仿真。工具箱提供了一个新的 LTI 观测器，方便了用户对系统的各种操作。

3. 系统设计

可以对系统进行极点配置，状态观测器设计，以及 LQG 最优控制等，并支持系统可控、可观标准实现、系统的最小实现、降阶实现、输入时延的 Pade 估计以及均衡实现等。

3.1 LTI 系统模型及转换

3.1.1 LTI 模型

LTI 模型可以指定为如下形式：

1. 传递函数模型(TF)

例如：

$$P(s)=\frac{s+2}{s^2+s+10}$$

2. 零极点增益模型(ZPK)

零极点增益模型是传递函数的一种特殊的形式，它通过零点、极点和增益相乘来表示系统。对于连续时间的 SISO 系统的零极点增益模型为

$$h(s)=k\frac{(s-z_1)\cdots(s-z_m)}{(s-p_1)\cdots(s-p_n)}$$

其中 k 为系统增益，z_i，$i=1,2,\cdots,m$ 为系统零点，p_i，$i=1,2,\cdots,n$ 为系统极点。对于离散时间系统其函数类似。

具体的例子如：

$$H(z)=\left[\frac{2(z-0.5)}{z(z+0.1)}\ \frac{(z^2+z+1)}{(z+0.2)(z+0.1)}\right]$$

3. 状态空间模型(SS)

状态空间模型通过线性微分或差分方程来描述系统的特性。对于连续时间模型，具有如下形式：

$$\frac{\mathrm{d}x}{\mathrm{d}t}=Ax+Bu$$

$$y=Cx+Du$$

A，B，C，D 为适当维数的系数矩阵，

x 为状态向量，u 和 y 分别为输入输出向量。

4. 频率响应数据模型(FRD)

FRD 由系统频率响应的采样组成。例如，可以将实验采集的频率响应数据存储在 FRD 模型中。

3.1.2 LTI 对象及其属性

在控制系统工具箱中将 LTI 系统的各种模型封装在一个对象中，从而可以方便的处理一个系统。

下面介绍一下 LTI 对象。LTI 对象可以为以下 4 种对象之一：ss 对象，tf 对象，zpk 对象，frd 对象。分别和状态空间模型、传递函数模型、零极点模型、频率响应数据模型相对应。这些不同的对象可以有共同的属性，也有各自的一些特殊属性。具体属性见表 3-1。

表 3-1 LTI 对象共有属性

属 性 名	描 述	值 的 类 型
ioDelay	I/O 时延	矩阵
InputDelay	输入时延	向量
InputGroup	输入对象数组	对象单元阵列
InputName	输入对象名	字符串数组单元阵列
Notes	模型的描述	文本
OutputDelay	输出时延	向量
OutputGroup	输出对象数组	对象单元阵列
OutputName	输出对象名	字符串数组单元阵列
Ts	采样周期	数
Userdata	附加数据	任意类型

tf 对象特有的属性见表 3-2。

表 3-2　对象 tf 特有的属性列表

属 性 名	描 述	值 的 类 型
den	传递函数分子系数	实行向量组成的单元阵列
num	传递函数分母系数	实行向量组成的单元阵列
Variable	传递函数变量	's', 'p', 'z', 'q', or 'z^-1'之一

zpk 对象特有的属性见表 3-3。

表 3-3　对象 zpk 特有的属性列表

属 性 名	描 述	值 的 类 型
k	增益	2 维实矩阵
p	极点	数值列向量组成的单元阵列
Variable	传递函数变量	's', 'p', 'z', 'q', or 'z^-1'之一
z	零点	数值列向量组成的单元阵列

ss 对象特有的属性见表 3-4。

表 3-4　对象 ss 特有的属性列表

属 性 名	描 述	值 的 类 型
a	系数 A	2 维实矩阵
b	系数 B	2 维实矩阵
c	系数 C	2 维实矩阵
d	系数 D	2 维实矩阵
e	系数 E	2 维实矩阵
Nx	系统状态总数	整数
StateName	状态变量名	字符串单元向量

frd 对象特有的属性见表 3-5。

表 3-5　对象 frd 特有的属性列表

属 性 名	描 述	值 的 类 型
Frequency	频率数据点	实向量
ResponseData	频率响应	复值多维阵列
Units	频率单位	字符串 'rad/s' 或 'Hz'

3.1.3　LTI 模型函数

MATLAB 控制系统工具箱提供了许多模型的建立函数和模型间的相互转换函数，见表 3-6。

表 3-6 模型建立及转换函数

函 数 名	函 数 功 能 描 述
dss	生成状态空间模型
filt	生成 DSP 形式的离散传递函数
frd	生成频率响应数据模型
frdata	获得频率响应数据模型数据
get	获取 LTI 对象的属性值
set	设置或修改 LTI 对象的属性值
ss	生成状态空间模型或者转换成状态空间模型
ssdata,dssdata	获得状态空间模型的数据
tf	生成传递函数模型或者转成传递函数模型
tfdata	获得传递函数模型数据
zpk	生成零极点模型或者转换成零极点模型
zpkdata	获得零极点模型数据

1. dss

功能：生成系统的描述状态空间（descriptor state-space）模型。

语法：
```
sys = dss(a,b,c,d,e)
sys = dss(a,b,c,d,e,Ts)
sys = dss(a,b,c,d,e,ltisys)
sys = dss(a,b,c,d,e,'Property1',Value1,...,'PropertyN',ValueN)
sys = dss(a,b,c,d,e,Ts,'Property1',Value1,...,'PropertyN',ValueN)
```

说明：sys = dss(a,b,c,d,e)生成连续系统的描述状态空间（descriptor state-space）模型：

$$E\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

其中 E 必须非奇异矩阵。a，b，c，d 分别对应于系统的 A，B，C，D 参数矩阵。函数返回一个 ss 对象。

sys = dss(a,b,c,d,e,Ts)生成离散生成连续系统的描述状态空间（descriptor state-space）模型：

$$Ex[n+1] = Ax[n] + Bu[n]$$

$$y[n] = Cx[n] + Du[n]$$

其中 E 必须非奇异矩阵。a，b，c，d 分别对应于系统的 A，B，C，D 参数矩阵。函数返回一个 ss 对象。Ts 为采样周期。

sys = dss(a,b,c,d,e,ltisys)生成的 ss 对象的其他属性继承 ltisys 对象的属性（包括采样周期）。

sys = dss(a,b,c,d,e,'Property1',Value1,...,'PropertyN',ValueN)和 sys = dss(a,b,c,d,e,Ts,'Property1',Value1,...,'PropertyN',ValueN)用来定义 ss 对象的属性值，属性名为'Property1'等，属性值为 Value1 等。关于属性值详见 3.1.2 节。

【例 1】

```
sys = dss(1,2,3,4,5,'td',0.1,'inputname','voltage',... 'notes','Just an example')
```

以上命令将生成如下模型：

$$5\dot{x} = x + 2u$$
$$y = 3x + 4u$$

2. filt

功能：生成 DSP 形式的离散传递函数。

语法：sys = filt(num,den)

sys = filt(num,den,Ts)

sys = filt(M)

sys = filt(num,den,'Property1',Value1,...,'PropertyN',ValueN)

sys = filt(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)

说明：在数字信号处理（DSP）中常常将传递函数以 z^{-1} 的形式表示出，例如：

$$H(z^{-1}) = \frac{2 + z^{-1}}{1 + 0.4z^{-1} + 2z^{-2}}$$

函数 filt 以一种简便的方法生成 DSP 格式的传递函数模型。

sys = filt(num,den)生成 DSP 形式的离散传递函数模型。num 为系统系数分子，den 为系统的系数分母，函数返回一个 tf 对象。采样周期未指明（sys.Ts= -1）。

sys = filt(num,den,Ts)特别指出采样周期为 Ts，单位为秒。

sys = filt(M)定义一个增益为 M 的静态系统。

sys = filt(num,den,'Property1',Value1,...,'PropertyN',ValueN)和 sys =filt(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)用以定义 tf 对象的属性值。关于属性值详见 3.1.2 节。

注意：函数 filt 和 tf 功能相同，只是将 Variable 属性设置为：z^{-1} 或者 q。

【例 2】

```
num = {1 , [1 0.3]}
den = {[1 1 2] ,[5 2]}
H = filt(num,den,'inputname',{'channel1' 'channel2'})
```

以上命令将生成如下模型：

$$H(z^{-1}) = \begin{bmatrix} \dfrac{1}{1 + z^{-1} + 2z^{-2}} & \dfrac{1 + 0.3z^{-1}}{5 + 2z^{-1}} \end{bmatrix}$$

3. frd

功能：生成频率响应数据模型或将其他模型转换为频率响应数据模型。

语法：sys = frd(response,frequency)

sys = frd(response,frequency,Ts)

sys = frd

sys = frd(response,frequency,ltisys)

sysfrd = frd(sys,frequency)

sysfrd = frd(sys,frequency,'Units',units)

说明：sys = frd(response,frequency)从储存在多维矩阵 response, frequency 中的数据生成 FRD 模型。

sys = frd(response,frequency,Ts)指明用 Ts 作采样周期生成 FRD 模型。

sys = frd 生成一个空的 FRD 模型。

sys = frd(response,frequency,ltisys)指定 FRD 模型的属性从 ltisys 中继承。

sysfrd = frd(sys,frequency)将 TF，SS，ZPK 模型转换为 FRD 模型，并指定在 frequency 处计算频率响应。

sysfrd = frd(sys,frequency,'Units',units)将 TF，SS，ZPK 模型转换为 FRD 模型，并指定频率单位为 'rad/s' 或 'Hz'。

【例 3】

```
freq = logspace(1,2);
resp = .05*(freq).*exp(i*2*freq);
sys = frd(resp,freq)
```

生成一个 FRD 模型。

4．frdata

功能：获得频率响应模型的数据。

语法：[response,freq] = frdata(sys)

[response,freq,Ts] = frdata(sys)

[response,freq] = frdata(sys,'v')

说明：[response,freq] = frdata(sys)返回 FRD 模型的响应数据和采样频率。假设 FRD 模型在频率 Nf 有 Nu 个 输入，Ny 个输出，则 response 为 Ny*Nu*Nf 维的矩阵，freq 为长度为 Nf 的列向量；

[response,freq,Ts] = frdata(sys) 返回 FRD 模型的响应数据和采样频率，并同时返回采样周期：Ts；

[response,freq] = frdata(sys,'v')对 SISO（单输入输出系统）强制返回值为列向量的形式。

【例 4】

```
freq = logspace(1,2,2);
resp = .05*(freq).*exp(i*2*freq);
sys = frd(resp,freq);
[resp,freq] = frdata(sys,'v')
```

以上命令将返回如下模型：

```
resp =
    0.2040 + 0.4565i
    2.4359 - 4.3665i
freq =
     10
     100
```

5．get

功能：获取 LTI 对象的属性值。

语法：Value = get(sys,'PropertyName')

get(sys)

Struct = get(sys)

说明：Value = get(sys,'PropertyName')获取 LTI 对象 sys 的 PropertyName 的属性值 Value。PropertyName 可以为属性名的全名（如：'UserData'），也可为无歧义的简写（如：'user'）。

get(sys)获取对象 sys 的所有属性值。

Struct = get(sys) 获取对象 sys 的所有属性值，并存如 MATLAB 的标准结构变量 Struct 中。属性名作为机构的域名，属性值作为结构的域值。

【例 5】 考虑如下离散时间的 SISO 转移函数。

```
h = tf(1,{1 2},0.1,'inputname','voltage','user','hello')
```

可以用如下的代码显示该对象的属性值：

```
get(h)
```

返回：

```
num = {[0 1]}
den = {[1 2]}
Variable = 'z'
Ts = 0.1
InputDelay = 0
OutputDelay = 0
ioDelay = 0
InputName = {'voltage'}
OutputName = {''}
InputGroup = {0x2 cell}
OutputGroup = {0x2 cell}
Notes = {}
UserData = 'hello'
```

6. set

功能：设置或修改 LTI 对象的属性值。

语法：set(sys,'Property',Value)

set(sys,'Property1',Value1,'Property2',Value2,...)

set(sys,'Property')

set(sys)

说明：set(sys,'Property',Value)设置 LTI 对象 sys 的 Property 属性为属性值 Value；

set(sys,'Property1',Value1,'Property2',Value2,...)设置多个属性值；

set(sys,'Property')显示 LTI 对象 sys 的 Property 属性的可能的属性值；

set(sys)显示 LTI 对象 sys 的所有属性及其可能的属性值。

【例 6】 考虑 sys = ss(1,2,3,4);生成的 SISO 状态空间模型。

通过 set(sys,'inputd',0.1,'inputn','torque','d',0,'user',dcgain(sys))

设置属性后，使用 get 查看设置的结果：

```
get(sys)
```

结果为：

```
a = 1
b = 2
c = 3
d = 0
e = []
Nx = 1
StateName = {''}
Ts = 0
InputDelay = 0.1
OutputDelay = 0
ioDelay = 0
InputName = {'torque'}
OutputName = {''}
InputGroup = {0x2 cell}
OutputGroup = {0x2 cell}
Notes = {}
UserData = -6
```

7．ss

功能：生成状态空间模型或者转换成状态空间模型。

语法：

```
sys = ss(a,b,c,d)
sys = ss(a,b,c,d,Ts)
sys = ss(d)
sys = ss(a,b,c,d,ltisys)
sys = ss(a,b,c,d,'Property1',Value1,...,'PropertyN',ValueN)
sys = ss(a,b,c,d,Ts,'Property1',Value1,...,'PropertyN',ValueN)
sys_ss = ss(sys)
sys_ss = ss(sys,'minimal')
```

说明：sys = ss(a,b,c,d)生成连续系统的状态空间模型

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

a，b，c，d 分别对应于 A，B，C，D 参数矩阵。

sys = ss(a,b,c,d,Ts)生成离散系统的状态空间模型

$$x[n+1] = Ax[n] + Bu[n]$$
$$y[n] = Cx[n] + Du[n]$$

a，b，c，d 分别对应于 A，B，C，D 参数矩阵。

sys = ss(a,b,c,d,ltisys)生成对象的其他属性从 ltisys 中继承（包括采样周期）。

sys = ss(a,b,c,d,'Property1',Value1,...,'PropertyN',ValueN)和 sys = ss(a,b,c,d,Ts,'Property1',

Value1,...,'PropertyN',ValueN)用来设置对象的属性。关于 ss 的属性详见 3.1.2 节。

sys_ss = ss(sys)将任意 TF 或 ZPK 模型转换为状态空间模型。

sys_ss = ss(sys,'minimal')该函数等价于 sys_ss = minreal(ss(sys))。

【例 7】 计算下面传递函数的状态空间的实现：

$$H(s)=\begin{bmatrix}\dfrac{s+1}{s^3+3s^2+3s+2}\\ \dfrac{s^2+3}{s^2+s+1}\end{bmatrix}$$

```
H = [tf([1 1],[1 3 3 2]) ; tf([1 0 3],[1 1 1])];
sys = ss(H)
size(sys)
```

输出为：

```
a =
          x1     x2     x3     x4     x5
   x1     -3    -1.5    -1      0      0
   x2      2      0      0      0      0
   x3      0      1      0      0      0
   x4      0      0      0     -1    -0.5
   x5      0      0      0      2      0
b =
        u1
   x1    1
   x2    0
   x3    0
   x4    2
   x5    0
 c =
          x1     x2     x3     x4     x5
   y1      0    0.5    0.5      0      0
   y2      0      0      0   -0.5    0.5
d =
       u1
   y1   0
   y2   1
Continuous-time model.
State-space model with 2 outputs, 1 input, and 5 states
```

8．ssdata,dssdata

功能：ssdata 获取传递函数模型数据；dssdata 获取状态空间模型数据。

语法：[a,b,c,d] = ssdata(sys)

[a,b,c,d,Ts] = ssdata(sys)

和

[a,b,c,d,e] = dssdata(sys)

[a,b,c,d,e,Ts] = dssdata(sys)

说明：返回值分别对应于获取对象 sys 的系数矩阵和采样周期。

9. tf

功能：生成传递函数模型或将 LTI 模型转化为传递函数模型。

语法：
```
sys = tf(num,den)
sys = tf(num,den,Ts)
sys = tf(M)
sys = tf(num,den,ltisys)
sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)
sys = tf(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)
sys = tf('s')
sys = tf('z')
tfsys = tf(sys)
tfsys = tf(sys,'inv')      % 仅适用于状态空间模型。
```

说明：sys = tf(num,den)生成连续时间系统的传递函数模型。num 和 den 分别为分子和分母。返回 sys 为 tf 对象。

注意：此时系数必须按照 z 的降幂排列。

sys = tf(num,den,Ts) 生成离散时间系统的传递函数模型。num 和 den 分别为分子和分母，Ts 为采样周期。返回 sys 为 tf 对象。

sys = tf(M)生成一个稳定增益为 M 的系统。

sys = tf(num,den,ltisys)生成 tf 对象的其他属性从 ltisys 继承（包括采样周期）。

sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)和 sys = tf(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)设置对象的属性值。关于对象属性详见属性详见 3.1.2 节。

sys = tf('s')指明使用 Laplace 变量 s 生成一个 TF 模型。

sys = tf('z'，Ts)使用采样周期 Ts 和离散变量 z 生成一个 TF 模型。

tfsys = tf(sys)将任意的 LTI 对象转换为传递函数的形式。缺省使用 zero 将状态空间模型转换成传递函数模型。

tfsys = tf(sys,'inv')使用状态空间的转置公式将状态空间模型转换成传递函数模型。

【例 8】 生成一个 2 输出/1 输入的传递函数：

$$H(p)=\begin{bmatrix}\dfrac{p+1}{p^2+2p+2}\\ \dfrac{1}{p}\end{bmatrix}$$

输入为 current，输出为 torque 和 ang.velocity。

使用如下命令：

```
num = {[1 1] ; 1};
den = {[1 2 2] ; [1 0]};
H = tf(num,den,'inputn','current','outputn',{'torque' 'ang. velocity'},'variable','p')
```

则系统返回：

```
Transfer function from input "current" to output...
                    p + 1
 torque:    ------------------
             p^2 + 2 p + 2

                  1
 ang.velocity:   ---
                  p
```

10. tfdata

功能：获得传递函数模型数据。

语法：[num,den] = tfdata(sys)

[num,den] = tfdata(sys,'v')

[num,den,Ts] = tfdata(sys)

说明：[num,den] = tfdata(sys)获得 sys 对象的传递函数的分子和分母多项式系数。num 和 den 分别为分子和分母。num(i,j)和 den(i,j)分别为系统第 j 个输出到第 i 个输入的传递函数的分子、分母多项式系数向量。如果 sys 为 ss 对象或 zpk 对象则先首先将其转化为 tf 对象。

[num,den] = tfdata(sys,'v')指定返回值为行向量的形式，仅使用于 SISO 系统。

[num,den,Ts] = tfdata(sys)同时返回系统的采样周期 Ts。

【例 9】

```
h = tf([1 1],[1 2 5]);
[num,den] = tfdata(h,'v')
```

则返回：

```
num =
      0     1     1
den =
1     2     5
```

11. zpk

功能：生成零极点模型或者将 LTI 模型转换为零极点增益模型。

语法：sys = zpk(z,p,k)

sys = zpk(z,p,k,Ts)

sys = zpk(M)

sys = zpk(z,p,k,ltisys)

sys = zpk(z,p,k,'Property1',Value1,...,'PropertyN',ValueN)

sys = zpk(z,p,k,Ts,'Property1',Value1,...,'PropertyN',ValueN)

```
sys = zpk('s')
sys = zpk('z')
zsys = zpk(sys)
zsys = zpk(sys,'inv')      % 仅适用于状态空间模型。
```

说明：sys = zpk(z,p,k)生成连续时间系统的零极点增益模型。其中 z、p、k 分别对应于系统的零点、极点和增益。返回一个 zpk 对象。

sys = zpk(z,p,k,Ts) 生成离散时间系统的零极点增益模型。其中 z、p、k 分别对应于系统的零点、极点和增益，Ts 为系统采样周期。返回一个 zpk 对象。

sys = zpk(M)定义一个增益为 M 的静态系统。

sys = zpk(z,p,k,ltisys)生成的 zpk 对象的其他属性从 ltisys 继承。

sys=zpk(z,p,k,'Property1',Value1,...,'PropertyN',ValueN) 和 sys = zpk(z,p,k,Ts,'Property1', Value1,...,'PropertyN',ValueN)定义 zpk 对象的属性值。关于对象属性详见属性详见 3.1.2 节。

sys = zpk('s')指明使用 Laplace 变量 s 生成一个 ZPK 模型。

sys = zpk('z'，Ts)使用采样周期 Ts 和离散变量 z 生成一个 ZPK 模型。

zsys = zpk(sys)将其他 LTI 对象转换为零极点增益模型。

zsys = zpk(sys,'inv') 使用状态空间的转置公式将状态空间模型转换成零极点增益模型。

【例 10】 考虑如下零极点增益模型

$$H(z)=\begin{bmatrix}\dfrac{1}{z-0.3}\\ \dfrac{2(z+0.5)}{(z-0.1+j)(z-0.1-j)}\end{bmatrix}$$

```
z = {[] ; -0.5}
p = {0.3 ; [0.1+i 0.1-i]}
k = [1 ; 2]
H = zpk(z,p,k,-1)
```

12. zpkdata

功能：获取零极点模型数据。

语法：
```
[z,p,k] = zpkdata(sys)
[z,p,k] = zpkdata(sys,'v')
[z,p,k,Ts,Td] = zpkdata(sys)
```

说明：[z,p,k] = zpkdata(sys)获取对象 sys 的零点 z，极点 p 和增益 k。其中 z 和 p 为单元阵列，k 为矩阵。z(i,j)和 k(i,j)定义了从第 j 个输入到第 i 个输出的零点、极点和增益。如果 sys 是一个 ss 对象或者 tf 对象，则首先将其转换为 zpk 对象。

[z,p,k] = zpkdata(sys,'v')仅仅用于 SISO 系统。其零点和极点返回值不再是单元阵列，而是以行向量返回。k 为一标量。

[z,p,k,Ts,Td] = zpkdata(sys)同时返回系统的采样周期 Ts 和输入延时 Td，单位为秒。对于连续时间系统，Td 为与所有输入通道一一对应的延时向量；对于离散时间系统，Td 为空。

【例 11】

```
H = zpk({[0];[-0.5]},{[0.3];[0.1+i 0.1-i]},[1;2],-1)
```

输出为：

```
Zero/pole/gain from input to output...
        1
 #1:  -------
       (z-0.3)
         2 (z+0.5)
 #2:  -------------------
       (z^2 - 0.2z + 1.01)
Sampling time: unspecified
```

3.1.4　模型检测函数

控制系统工具箱还提供了一组模型检测函数，这类函数格式和功能比较简单，这里给出其函数格式，见表 3-7。

表 3-7　模型检测函数

函　数	语　法	功　能
class	str = class(object)	显示模型类型('tf', 'zpk', 'ss', or 'frd')
hasdelay	hasdelay(sys)	判断系统是否有任何类型的时延。是则返回 1；否，返回 0
isa	K = isa(obj,'class_name')	判断 obj 是否为 class_name 类。是返回 1；否，返回 0
isct	boo = isct(sys)	判断 LTI 对象 sys 是否为连续时间系统。是则返回 1；否，返回 0
isdt	boo = isdt(sys)	判断 LTI 对象 sys 是否为离散时间系统。是则返回 1；否，返回 0
isempty	boo = isempty(sys)	判断 LTI 对象 sys 是否为空。是则返回 1；否，返回 0
isproper	boo = isproper(sys)	判断 LTI 对象 sys 是否为正则。是则返回 1；否，返回 0
issiso	boo = issiso(sys)	判断 LTI 对象 sys 是否为单输入输出系统（SISO）。是则返回 1；否，返回 0
ndims	n = ndims(sys)	返回 LTI 模型或 LTI 数列的维数
size	（见下面详细说明）	

- size

功能：计算 LTI 模型输入、输出的维数，TF、SS 和 ZPK 模型的阶次，FRD 模型的频率数。

语法：
```
size(sys)
d = size(sys)
Ny = size(sys,1)
Nu = size(sys,2)
Sk = size(sys,2+k)
Ns = size(sys,'order')
Nf = size(sys,'frequency')
```

说明：size(sys)返回单个 LTI 模型的输入输出维数。

d = size(sys):

- 当系统为单个 LTI 模型时，返回行向量 d = [Ny Nu]，Ny 为输出维数，Nu 为输入维数；
- 当系统为由 LTI 模型组成的 S1*S2*….*Sp 维的矩阵时，返回 d = [Ny Nu S1 S2 ... Sp]，Ny 为输出维数，Nu 为输入维数；

Ny = size(sys,1)返回系统 sys 的输出维数；

Nu = size(sys,2) 返回系统 sys 的输入维数；

Sk = size(sys,2+k)当系统为 LTI 模型数列时，返回第 k 个模型的维数；

Ns = size(sys,'order') 返回 TF、SS 和 ZPK 模型的阶次，当 sys 为 LTI 数组时，返回该数组中最大的模型阶次；

Nf = size(sys,'frequency')返回 FRD 模型的频率数。

3.2 状态空间的实现

3.2.1 状态空间的实现

状态空间的实现包括以下几个要点：

1. 相似变换

系统 LTI 的状态方程由

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

给出，现引进非奇异变换矩阵 T，则系统变换后为

$$\bar{A} = TAT^T, \qquad \bar{B} = TB, \qquad \bar{C} = CT^T$$

该变换为相似变换，相似变换步影响系统的特征参数和系统的结构。

2. 系统的可观性、可观标准及系统可观和不可观分解

系统的可观性可以通过 Gram 矩阵判别或通过秩判据来判定。

对于连续时间系统，Gram 矩阵判据为

（C，A）为可观的当且仅当 Gram 矩阵

$$W_o = \int_0^\infty e^{A\tau} C^T C e^{A\tau} d\tau$$

为正定的。

3. 系统的可控性、可控标准及系统可控和不可控分解

系统的可控性可以通过 Gram 矩阵判别或通过秩判据来判定。

对于连续时间系统，Gram 矩阵判据为

（A ，B）为可控的当且仅当 Gram 矩阵

$$W_c = \int_0^\infty e^{A\tau} BB^T e^{A\tau} d\tau$$

为正定的。

对于离散时间系统的可观性及可控性判定，则可通过离散形式的 Gram 来判定，对应于可观性和可控性的离散形式的 Gram 矩阵分别为

$$W_c = \sum_{k=0}^{\infty} A^k BB^T (A^T)^k \ , \qquad W_o = \sum_{k=0}^{\infty} (A^T)^k C^T C A^k$$

3.2.2　状态空间的实现的函数

MATLAB 控制系统工具箱提供的状态空间实现函数见表 3-8。

表 3-8　状态空间实现函数列表

函数名	函数功能描述
canon	状态空间的正则实现
ctrb	可控矩阵计算
ctrbf	系统的可控与不可控分解
gram	求系统的可控与可观 gramian 矩阵
obsv	可观矩阵计算
obsvf	系统的可观与不可观分解
ss2ss	相似变换
ssbal	状态空间的对角均衡实现

1．canon

功能：状态空间的正则实现。

语法：csys = canon(sys,'type')

[csys,T] = canon(sys,'type')

说明：canon 函数可计算连续时间或者离散时间 LTI 对象的正则实现。该函数支持两种正则形式：模态矩阵(modal matrix)和伴随矩阵(companion matrix)形式。

(1) 模态矩阵形式。设系统特征值为：

$$(\lambda_1, \sigma \pm j\omega, \lambda_2)$$

则模态矩阵 A 为

$$\begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \sigma & \omega & 0 \\ 0 & -\omega & \sigma & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}$$

(2) 伴随矩阵形式。设系统的特征多项式为

$$p(s) = s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n$$

则伴随矩阵为

$$A = \begin{bmatrix} 0 & 0 & .. & .. & 0 & -a_n \\ 1 & 0 & 0 & .. & 0 & -a_{n-1} \\ 0 & 1 & 0 & . & : & : \\ : & 0 & . & . & : & : \\ 0 & . & . & 1 & 0 & -a_2 \\ 0 & .. & .. & 0 & 1 & -a_1 \end{bmatrix}$$

csys = canon(sys,'type')计算 LTI 对象 sys 的正则实现。如果系统不是以状态空间模型给出，则首先进行模型转换。其中可以为以下字符串之一：

'modal'：计算模态矩阵正则实现；

'companion'：计算伴随矩阵正则实现；

[csys,T] = canon(sys,'type')同时返回相似变换矩阵 T。当给出的 LTI 对象不是 ss 对象时，T 为空矩阵。

注意：

- 模态实现要求系统的 A 矩阵为可对角化的。
- 伴随实现需要系统关于第一个输入为可控的，应尽量避免求伴随实现。

2. ctrb

功能：可控矩阵计算。

语法：Co = ctrb(A,B)

Co = ctrb(sys)

说明：ctrb 函数用于计算 LTI 系统的可控矩阵(controllability matrix)。对于一个 $n*n$ 的矩阵 A 及一个 $n*m$ 的矩阵 B。其可控矩阵为：

$$Co=\left[B\ AB\ A^2B\ldots A^{n-1}B\right]$$

如果 Rank(co)=n，则系统可控。

Co = ctrb(A,B)计算由矩阵 A 和矩阵 B 给出的系统的可控矩阵 Co。

Co = ctrb(sys)计算状态空间 LTI 对象的可控矩阵 Co。该调用等价于 Co = ctrb(sys.A,sys.B)。

【例 1】

```
A =
     1     1
     4    -2
B =
     1    -1
     1    -1
  Co=ctrb(A,B);
unco=length(A)-rank(Co)
```

MATLAB 返回：

```
unco =
     1
```

3. ctrbf

功能：系统的可控与不可控分解。

语法：[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C,tol)

说明：如果系统的阶为 n，而系统的可控矩阵的阶小于 n，则存在一个相似变换

$$\bar{A}=TAT^T,\qquad \bar{B}=TB,\qquad \bar{C}=CT^T$$

将系统(A，B，C)进行可控与不可控分解。使系统具有如下格式：

$$\bar{A}=\begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix},\quad \bar{B}=\begin{bmatrix} 0 \\ B_c \end{bmatrix},\quad \bar{C}=[C_{nc}\quad C_c]$$

则(A_C, B_C)构成系统的可控子空间。A_{UC}的所有特征值均是不可控的，并且有

$$C_c(sI-A_c)^{-1}B_c=C(sI-A)^{-1}B$$

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)将系统分解为可控/不可空两部分。如上所述，T 为相似变换，k 是长度为 n 的一个矢量，其元素为各个块的秩。sum(k)可求出 A 中可控部分的秩。(Abar,Bbar,Cbat)对应于转换后系统的(A，B，C)。

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C,tol)定义误差容限 tol，缺省时 10*n*norm(A,1)*eps。

【例 2】

```
A =
      1      1
      4     -2
B =
      1     -1
      1     -1
C =
      1      0
      0      1
[Abar,Bbar,Cbar,T,k]=ctrbf(A,B,C)
```

MATLAB 返回：

```
Abar =
   -3.0000       0
   -3.0000    2.0000
Bbar =
    0.0000    0.0000
    1.4142   -1.4142
Cbar =
   -0.7071    0.7071
    0.7071    0.7071
T =
   -0.7071    0.7071
    0.7071    0.7071
k =
1      0
```

系统分解表明，该系统有一个可控状态和一个不可控状态。

4．gram

功能：求系统的可控与可观 gramian 矩阵。

语法：Wc = gram(sys,'c')

Wo = gram(sys,'o')

说明：gram 函数可计算系统的可控于可观 gramian 矩阵。Gramian 矩阵可用于研究系统的可控于可观性。该函数较 ctrb 和 obsv 函数有更好的属性。对于

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

给出的连续时间系统，其可控 gramian 矩阵 Wc 与可观 gramian 矩阵 Wo 为：

$$W_c = \int_0^{\infty} e^{A\tau} BB^T e^{A\tau} d\tau$$

$$W_o = \int_0^{\infty} e^{A\tau} C^T C e^{A\tau} d\tau$$

对于

$$x[n+1] = Ax[n] + Bu[n]$$

$$y[n] = Cx[n] + Du[n]$$

给出的离散时间系统，其可控 gramian 矩阵 Wc 与可观 gramian 矩阵 Wo 为：

$$W_c = \sum_{k=0}^{\infty} A^k BB^T (A^T)^k \text{ ,} \qquad W_o = \sum_{k=0}^{\infty} (A^T)^k C^T CA^k$$

且可控 gramian 矩阵负定与系统可控等价，可观 gramian 矩阵负定与系统可观等价。

Wc = gram(sys,'c')计算 LTI 对象 sys 的可控 gramian 矩阵。

Wo = gram(sys,'o')计算 LTI 对象 sys 的可观 gramian 矩阵。

算法：Wc 可通过求解下述连续时间 Lyapunov 方程：

$$AW_c + W_c A^T + BB^T = 0$$

或者离散时间 Lyapunov 方程：

$$AW_c A^T - W_c + BB^T = 0$$

Wo 可通过求解下述连续时间 Lyapunov 方程：

$$A^T W_o - W_o A + C^T C = 0$$

或者离散时间 Lyapunov 方程：

$$A^T W_o A - W_o + C^T C = 0$$

5. obsv

功能：可观矩阵计算。

语法：Ob = obsv(A,B)

Ob = obsv(sys)

说明：obsv 函数用于计算 LTI 系统的可观矩阵(observability matrix)。对于一个 n×n 的矩阵 A 及一个 p×n 的矩阵 B。其可观矩阵为：

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

如果 Rank(Ob)=n，则系统可控。

Ob = obsv(A,B)计算由矩阵 A 和矩阵 B 给出的系统的可观矩阵 Ob。Ob = obsv(sys)计算状态空间 LTI 对象的可观矩阵 Ob。该调用等价于 Ob =obsv(sys.A,sys.C)。

【例 3】

```
A =
      1     1
      4    -2
C =
      1     0
      0     1
Ob = obsv(A,C);
unob = length(A)-rank(Ob)
```

MATLAB 返回：

```
unob =
      0
```

6. obsvf

功能：系统的可观与不可观分解。

语法：[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)

　　　[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C,tol)

说明：如果系统的阶为 n，而系统的可观矩阵的阶小于 n，则存在一个相似变换：

$$\bar{A} = TAT^T, \quad \bar{B} = TB, \quad \bar{C} = CT^T$$

将系统(A，B，C)进行可观与不可观分解。使系统具有如下格式：

$$\bar{A} = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix}, \quad \bar{C} = [0 \quad C_o]$$

则（C_0,A_0）构成系统的可观子空间。

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)将系统分解为可观/不可观两部分。如上所述，T 为相似变换，k 是长度为 n 的一个矢量，其元素为各个块的秩。sum(k)可求出 A 中可观部分的秩。(Abar,Bbar,Cbat)对应于转换后系统的(A,B,C)。

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C,tol)定义误差容限 tol，缺省时 10*n*norm(a,1)*eps。

【例 4】

```
A =
      1     1
      4    -2
B =
      1    -1
      1    -1
C =
      1     0
```

```
        0        1
[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)
MATLAB 返回：
Abar =
        1        1
        4       -2
Bbar =
        1        1
        1       -1
Cbar =
        1        0
        0        1
T =
        1        0
        0        1
k =
        2        0
```

7. ss2ss

功能：相似变换。

语法：sysT = ss2ss(sys,T)

说明：ss2ss 可完成系统的相似变换。相似变换即对于如下系统：

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

进行 $\bar{x} = Tx$ 变换，变换后的系统为

$$\dot{\bar{x}} = TAT^{-1}\bar{x} + TBu$$
$$y = CT^{-1}\bar{x} + Du$$

sysT = ss2ss(sys,T)完成上述状态空间 LTI 对象 sys 的相似变换。其中，T 为变换矩阵。

8. ssbal

功能：状态空间的对角均衡实现。

语法：[sysb,T] = ssbal(sys)

[sysb,T] = ssbal(sys,condT)

说明：ssbal 将系统(A,B,C)进行相似变换，变换矩阵 T 具有下述形式：

$$\begin{bmatrix} TAT^{-1} & TB/\alpha \\ \alpha CT^{-1} & 0 \end{bmatrix}$$

该矩阵的行和列具有近似相等的范数。从而得到系统的均衡实现

$$(TAT^{-1}, TB/\alpha, \alpha CT^{-1}, D)$$

[sysb,T] = ssbal(sys)计算状态空间的均衡实现。

[sysb,T] = ssbal(sys,condT)同时定义相似转换矩阵 T 的范数上界 condT。缺省值为 condT=Inf。

【例 5】 实现下述系统的均衡实现。

$$A=\begin{bmatrix}1 & 10^4 & 10^2\\ 0 & 10^2 & 10^5\\ 10 & 1 & 0\end{bmatrix},\quad B=\begin{bmatrix}1\\1\\1\end{bmatrix},\quad C=[0.1\quad 10\quad 100]$$

```
a = [1 1e4 1e2;0 1e2 1e5;10 1 0];
b = [1;1;1];
c = [0.1 10 1e2];
sys = ss(a,b,c,0)
ssbal(sys)
```

MATLAB 返回：

```
a =
              x1        x2          x3
       x1      1      2500     0.39063
       x2      0       100      1562.5
       x3   2560        64           0
b =
            u1
       x1   0.125
       x2   0.5
       x3    32
c =
             x1          x2        x3
       y1    0.8         20     3.125
d =
           u1
       y1        0
       Continuous-time system.
```

3.3　系统时域响应

3.3.1　系统时域响应

对控制系统，其模型均由微分方程或差分方程给出，因此通过从给定的初始值出发，通过某种算法逐步求出系统在每个时刻的响应，从而可以实现对系统的分析。对于 LTI 系统，MATLAB 控制系统工具箱提供了许多函数对系统进行仿真，见表 3-9。

表 3-9 系统时域响应函数列表

函数名	函数功能描述
gensig	输入信号产生
impulse	计算系统脉冲响应
Initial	计算系统零输入响应
lsim	对系统的任意输入进行仿真
step	计算系统阶跃响应

1. gensig

功能：输入信号产生。

语法：[u,t] = gensig(type,tau)

[u,t] = gensig(type,tau,Tf,Ts)

说明：[u,t] = gensig(type,tau)产生一个类型为 type 的信号序列 u(t)。信号周期为 tau。其中 type 可以为以下类型标识字符串之一：

- 'sin'：正弦波；
- 'square'：方波；
- 'pulse'：脉冲序列。

[u,t] = gensig(type,tau,Tf,Ts)同时定义持续时间 Tf 和采样周期 Ts。

【例 1】 生成一个周期为 5s，持续时间为 30s，采样周期为 0.1s 的方波，如图 3-1 所示。

```
[u,t] = gensig('square',5,30,0.1);
plot(t,u)
axis([0 30 -1 2])
```

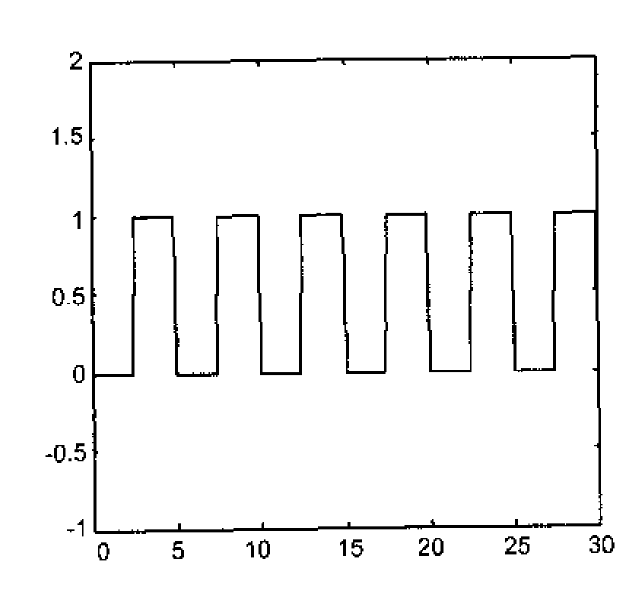


图 3-1 生成方波输入曲线

2. impulse

功能：计算系统脉冲响应。

语法：impulse(sys)

```
impulse(sys,t)
impulse(sys1,sys2,...,sysN)
impulse(sys1,sys2,...,sysN,t)
impulse(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[y,t,x] = impulse(sys)
```

说明：impulse 函数用于计算系统的单位冲激响应。当调用无输出变量时，impulse 在当前图形窗口中直接绘出系统的单位冲激响应；

impulse(sys)计算并在当前窗口绘制 LTI 对象 sys 的脉冲响应，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统；

impulse(sys,t)定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配；

impulse(sys1,sys2,...,sysN)和 impulse(sys1,sys2,...,sysN,t)同时仿真多个 LTI 对象；

impulse(sys1,'PlotStyle1',...,sysN,'PlotStyleN')定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 MATLAB 标准命令 plot 支持的各种属性标识字符串；

[y,t] = impulse(sys)、[y,t,x] = impulse(sys) 、y = impulse(sys,t)计算仿真数据，且不在窗口显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹数据。

对于 MIMO 系统，y 的维数为：

（t 的长度）×（输出变量数）×（输入变量数）

y(:,:,j)对应于第 j 个通道；同样地，x 的维数为：

（t 的长度）×（输出变量数）×（输入变量数）

【例 2】 绘制如下系统的脉冲响应，如图 3-2 所示。

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

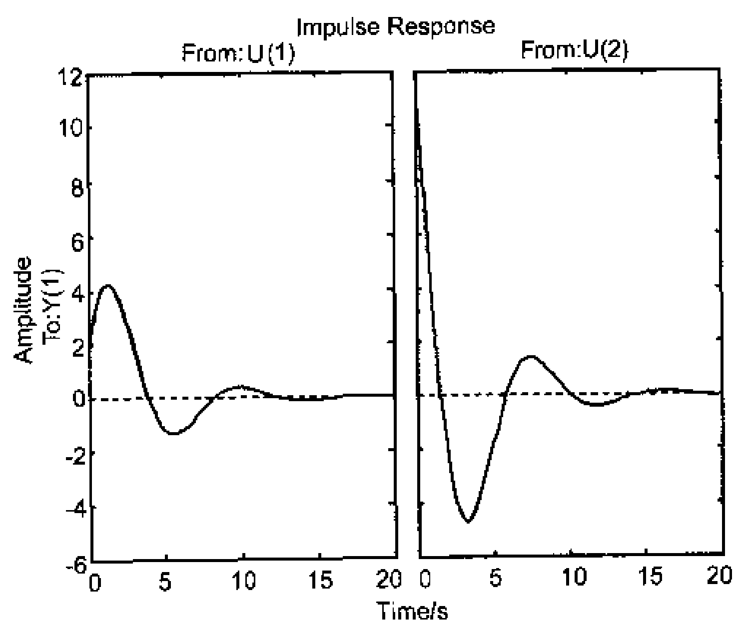


图 3-2　系统的脉冲响应

$$y=[1.9691\ \ 6.4493]\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

```
a = [-0.5572 -0.7814;0.7814   0];
b = [1 -1;0 2];
c = [1.9691   6.4493];
sys = ss(a,b,c,0);
impulse(sys)
```

3. initial

功能：计算系统零输入响应。

语法：initial(sys,x0)

```
initial(sys,x0,t)
initial(sys1,sys2,...,sysN,x0)
initial(sys1,sys2,...,sysN,x0,t)
initial(sys1,'PlotStyle1',...,sysN,'PlotStyleN',x0)
[y,t,x] = initial(sys,x0)
```

说明：initial 函数用于计算系统的零输入响应。当调用无输出变量时，initial 在当前图形窗口中直接绘出系统地单位冲激响应。

initial(sys,x0)计算并在当前窗口绘制 LTI 对象 sys 的零输入响应，可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

initial(sys,x0,t)定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配。

initial(sys1,sys2,...,sysN,x0)和 initial(sys1,sys2,...,sysN,x0,t)同时仿真多个 LTI 对象。

initial(sys1,'PlotStyle1',...,sysN,'PlotStyleN',x0) 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 MATLAB 标准命令 plot 支持的各种属性标识字符串。

[y,t,x] = initial(sys,x0)和[y,t,x] = initial(sys,x0,t)计算仿真数据，且不在窗口上显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹数据。

【例 3】 绘制如下系统的零输入响应，如图 3-3 所示。

$$\begin{bmatrix}\dot{x}_1\\\dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$$y=[1.9691\ \ 6.4493]\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

```
a = [-0.5572   -0.7814;0.7814   0];
c = [1.9691   6.4493];
x0 = [1 ; 0]
sys = ss(a,[],c,[]);
initial(sys,x0)
```

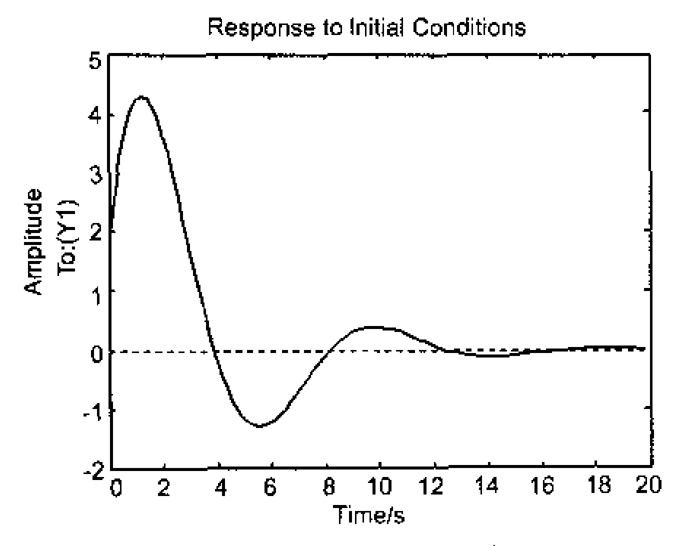


图 3-3 系统的零输入响应

4．lsim

功能：对系统的任意输入进行仿真。

语法：lsim(sys,u,t)

```
lsim(sys,u,t,x0)
lsim(sys,u,t,x0,'zoh')
lsim(sys,u,t,x0,'foh')
lsim(sys1,sys2,...,sysN,u,t)
lsim(sys1,sys2,...,sysN,u,t,x0)
lsim(sys1,'PlotStyle1',...,sysN,'PlotStyleN',u,t)
[y,t,x] = lsim(sys,u,t,x0)
```

说明：lsim 函数用于计算系统的任意输入响应。当调用无输出变量时，lsim 在当前图形窗口中直接绘出系统的单位冲激响应。

lsim(sys,u,t)计算并在当前窗口绘制 LTI 对象 sys 在输入为 u(t)时的响应，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

lsim(sys,u,t,x0)定义系统的初始状态 x0。

lsim(sys,u,t,x0,'zoh')和 lsim(sys,u,t,x0,'foh')定义了输入值应采用的插值方法。

lsim(sys1,sys2,...,sysN,u,t)和 lsim(sys1,sys2,...,sysN,u,t,x0)同时仿真多个 LTI 对象。

lsim(sys1,'PlotStyle1',...,sysN,'PlotStyleN',u,t) 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 MATLAB 标准命令 plot 支持的各种属性标识字符串。

[y,t] = lsim(sys,u,t)、[y,t,x] = lsim(sys,u,t)和[y,t,x] = lsim(sys,u,t,x0)计算仿真数据，且不在窗口上显示。

【例 4】 计算下述系统的方波响应。其中，方波的周期为 4s，持续时间为 10s，采样周期为 0.1s，如图 3-4 所示。

$$H(s)=\begin{bmatrix}\dfrac{2s^2+5s+1}{s^2+2s+3}\\[2mm]\dfrac{s-1}{s^2+s+5}\end{bmatrix}$$

```
[u,t] = gensig('square',4,10,0.1);
H = [tf([2 5 1],[1 2 3]) ; tf([1 -1],[1 1 5])]
lsim(H,u,t)
```

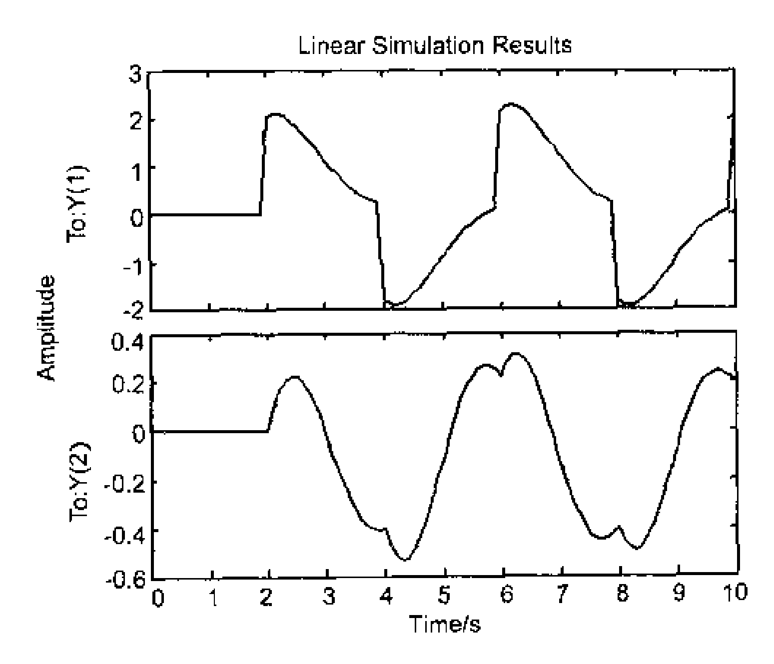


图 3-4 系统的方波响应

5. step

功能：计算系统阶跃响应。

语法：step(sys)

step(sys,t)

step(sys1,sys2,...,sysN)

step(sys1,sys2,...,sysN,t)

step(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[y,t,x] = step(sys)

说明：step 函数用于计算系统的阶跃响应。当调用无输出变量时，step 在当前图形窗口中直接绘出系统的阶跃响应。

step(sys)计算并在当前窗口绘制 LTI 对象 sys 的阶跃响应，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

step(sys,t) 定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配。

step(sys1,sys2,...,sysN)和 step(sys1,sys2,...,sysN,t) 同时仿真多个 LTI 对象。

step(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 MATLAB 标准命令 plot 支持的各种属性标识字符串。

[y,t,x] = step(sys) 计算仿真数据，且不在窗口显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹数据。对于 MIMO 系统，y 的维数为：

（t 的长度）×（输出变量数）×（输入变量数）



y(:,:,j)对应于第 j 个通道；同样地，x 的维数为：

（t 的长度）×（输出变量数）×（输入变量数）

【例 5】 绘制如下系统的脉冲响应，如图 3-5 所示。

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.5572 & -0.7814 \\ 0.7814 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$y = [1.9691 \quad 6.4493] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

```
a = [-0.5572    -0.7814;0.7814   0];
b = [1 -1;0 2];
c = [1.9691   6.4493];
sys = ss(a,b,c,0);
step(sys)
```

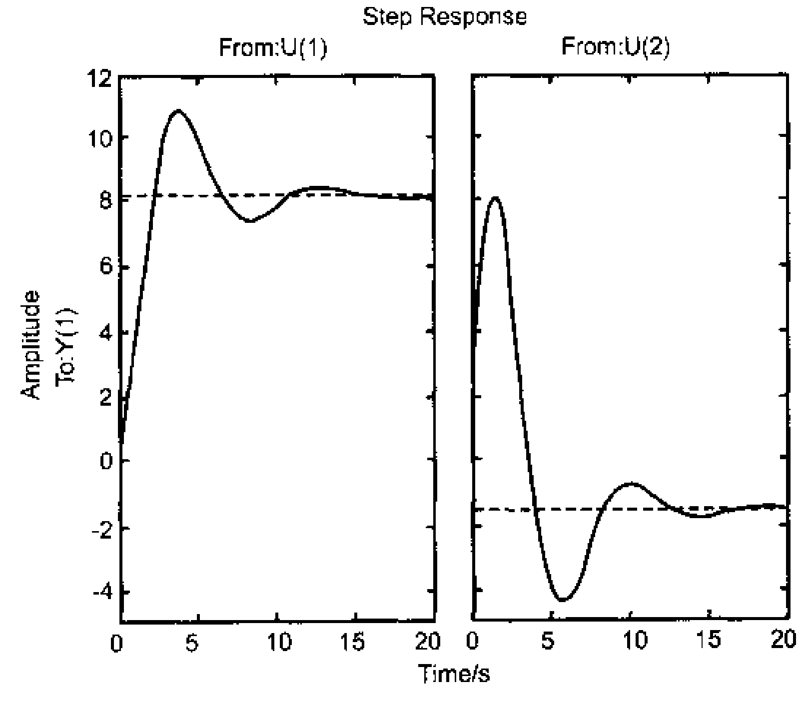


图 3-5　系统的脉冲响应

3.3.2　系统时域延迟

MATLAB 系统控制工具箱中提供的系统时域延迟函数见表 3-10。

表 3-10　系统时域延迟函数列表

函数名	函数功能描述
delay2z	转换离散时间模型或 FRD 模型的时延
pade	计算时延的 Padé 近似
totaldelay	提供 LTI 系统总共的时延

1. delay2z

功能：在 z=0 处将离散时间系统 SS、TF 或 ZPK 模型的时延替换为极点，或者将 FRD 模型的时延用相位变换代替。

语法：sys = delay2z(sys)

说明：sys = delay2z(sys)在 z=0 处将离散时间系统 SS、TF 或 ZPK 模型的所有时延映射为极点。特别的：采样周期为 k 的系统的时延用 (1/z)^k 代替。

对于 FRD 模型，delay2z 将所有的时延转换为频率响应数据，该函数对离散和连续 FRD 模型都适用。

【例 1】

```
z=tf('z',-1);
sys=(-.4*z -.1)/(z^2 + 1.05*z + .08)
sys.InputDelay = 1;
sys = delay2z(sys)
```

MATLB 输出为：

```
Transfer function:
   -0.4 z - 0.1
-------------------
z^2 + 1.05 z + 0.08
Sampling time: unspecified
Transfer function:
       -0.4 z - 0.1
-----------------------
z^3 + 1.05 z^2 + 0.08 z
Sampling time: unspecified
```

2. pade

功能：计算时延的 Padé 近似。

语法：[num,den] = pade(T,N)

pade(T,N)

sysx = pade(sys,N)

sysx = pade(sys,NI,NO,Nio)

说明：[num,den] = pade(T,N)以转移函数形式返回连续时间 I/O 时延的第 N 次 Padé 近似。行向量 num 和 den 分别包含分子和分母的系数，按照变量降阶的顺序排列。

pade(T,N)作 N 次 Padé 近似的相位响应图，并与具有时延 T 的 I/O 系统的响应比较。

sysx = pade(sys,N)生成连续时延系统 sys 的无时延近似 sysx。所有的时延用 N 次 Padé 近似。

sysx = pade(sys,NI,NO,Nio)为每个输入、输出和 I/O 时延指定独立的近似阶次。这些阶次在整数数列 NI，NO，Nio 中给出：

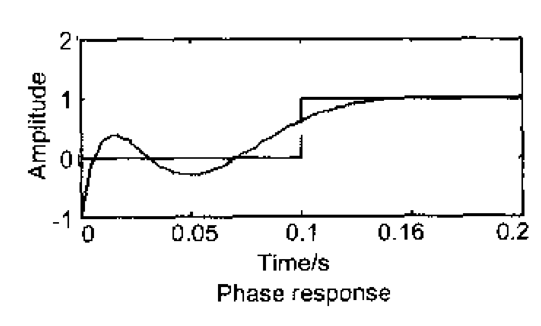
- NI(j)为第 j 个输入的近似阶次；
- NO(i)为第 i 个输出的近似阶次；



- Nio(i,j)为从输入 j 到输出 i 的 I/O 时延的近似阶次。

【例 2】 计算一个 0.1s I/O 时延的 3 次 Padé 近似，并比较真实时延和它近似的时间频率响应(图 3-6)。

pade(0.1,3)



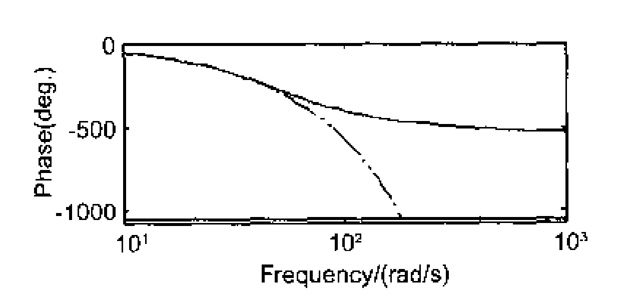


图 3-6 比较 pade 近似和真实时间频率响应

3. totaldelay

功能：返回 LTI 系统所有 I/O 时延。

语法：td = totaldelay(sys)

说明：td = totaldelay(sys) 返回 LTI 模型 sys 的所有组合的 I/O 时延。对于连续时间系统时延表示为秒，对于离散时间系统时延表示为采样周期的整数倍。返回 td 为矩阵，它综合了所有 InputDelay，OutputDelay 和 ioDelay 属性。

【例 3】

```
sys = tf(1,[1 0]);
sys.inputd = 2;
sys.outputd = 1.5;
td = totaldelay(sys)
```

MATLAB 返回：

```
td =
      3.5000
```

结果的 I/O 映射为：

$$e^{-2s}\times\frac{1}{s}e^{-1.5s}=e^{-3.5s}\frac{1}{s}$$

3.4 系统频率响应

MATLAB 系统控制工具箱种提供的关于系统频率响应曲线绘制及频域分析的函数见表 3-11。

表 3-11 系统频率响应函数列表

函数名	函数功能描述
allmargin	计算所有的交叉频率和稳定裕度
bode	计算并绘制 Bode 频率响应
bodemag	计算并绘制 Bode 频率响应振幅

(续)

函数名	函数功能描述
evalfr	计算系统单频率点的频率响应
freqresp	计算系统频率响应
interp	在 FRD 模型频率点间插入频率响应数据
linspace	生成平均频率间隔的向量
logspace	生成平均对数频率间隔的向量
margin	计算系统的增益和相角稳定裕度
ngrid	Nichols 网格线绘制
nichols	Nichols 绘制
nyquist	Nyquist 绘制
sigma	系统的奇异值 Bode 图绘制

1. allmargin

功能：计算所有的交叉频率和稳定裕度。

语法：S = allmargin(sys)

说明：allmargin 计算 SISO 开放循环模型 sys 的增益、相位、时延裕度和相应的交叉频率。allmargin 可以应用在任何 SISO 模型上，包括具有时延的模型。

输出 S 为一结构体，它具有如下的域：

- GMFrequency：所有-180 度的交叉频率。
- GainMargin：相应的增益裕度，定义为 1/G，G 为在交叉处的增益。
- PMFrequency：所有 0dB 的交叉频率。
- PhaseMargin：以角度表示的对应的相位增益。
- DMFrequency 和 DelayMargin：关键的频率和对应的时延裕度，在连续时间系统时，时延以秒的形式给出。在离散时间系统中，时延以采样周期的整数倍给出。
- Stable：如果闭循环系统稳定，则为 1；否则为 0。

2. bode

功能：Bode 图绘制。

语法：
```
bode(sys)
bode(sys,w)
bode(sys1,sys2,...,sysN)
bode(sys1,sys2,...,sysN,w)
bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[mag,phase,w] = bode(sys)
```

说明：bode 函数计算并显示系统的 Bode 图。当调用无输出变量时，bode 在当前图形窗口中进行 bode 图绘制。

bode(sys)计算并在当前窗口绘制 LTI 对象 sys 的 bode 图，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。绘制时的频率范围将依据系统的零极点决定。

bode(sys,w)显式定义绘制时的频率范围或者频率点 w。若要定义频率范围，w 必须具有

[wmin,wmax]格式；如果定义频率点，则 w 必须为由需要频率点频率组成的向量。

bode(sys1,sys2,...,sysN)和 bode(sys1,sys2,...,sysN,w)同时在一个窗口绘制多个 LTI 对象的 Bode 图。这些系统必须具有同样多的输入和输出数，但可以同时含有离散时间和连续时间系统。该调用常用于多个系统 Bode 图的比较。

bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 MATLAB 标准命令 plot 支持的各种属性标识字符串。

[mag,phase,w] = bode(sys)和[mag,phase] = bode(sys,w)计算 bode 图数据，且不在窗口显示图形。其中，mag 为 bode 图的幅度值，phase 为 bode 图的相位值，w 为 bode 图的频率点。Mag 和 phase 均为 3 维向量，其大小为

（输出变量数）×（输入变量数）×（w 的长度）

对于 SISO 系统，有

$\omega_k = W(k)$

mag(1,1,k)=$|h(j\omega_k)|$

phase(1,1,k)=$\angle h(j\omega_k)$

对于 MIMO 系统，第 i 个输入与第 j 个输出的传递函数 hij，有

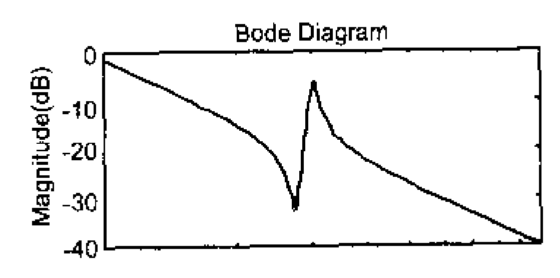
mag(i,j,k)=$|h_{ij}(j\omega_k)|$

phase(i,j,k)=$\angle h_{ij}(j\omega_k)$

【例 1】 绘制下述系统的 bode 图，如图 3-7 所示。

$$H(s)=\frac{s^2+0.1s+7.5}{s^4+0.12s^3+9s^2}$$

```
g = tf([1 0.1 7.5],[1 0.12 9 0 0]);
bode(g)
```



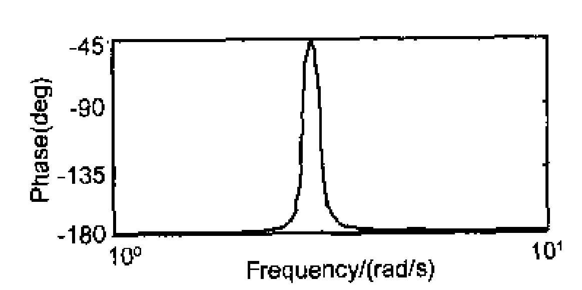


图 3-7 Bode 图

3. bodemag

功能：计算并绘制 Bode 频率响应的振幅。

语法：bodemag(sys)

bodemag(sys,{wmin,wmax})

bodemag(sys,w)

bodemag(sys1,sys2,...,sysN,w)

bodemag(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

说明：bodemag(sys)绘制 LTI 对象 sys 的频率响应的振幅。系统自动选择频率范围和所取的点。

bodemag(sys,{wmin,wmax})绘制频率在 wmin，wmax 之间的振幅，频率单位为：rad/sec。

bodemag(sys,w)使用用户定义的频率向量 w 作图。

bodemag(sys1,sys2,...,sysN,w)作多个系统的频率响应的振幅，频率向量 w 为可选参数。

bodemag(sys1,'PlotStyle1',...,sysN,'PlotStyleN')：对每个系统指定作图的颜色、线条等。

【例 2】 绘制系统波特响应曲线，如图 3-8 所示。

```
g = tf([1 0.1 7.5],[1 0.12 9 0 0]);
bodemag(g)
```

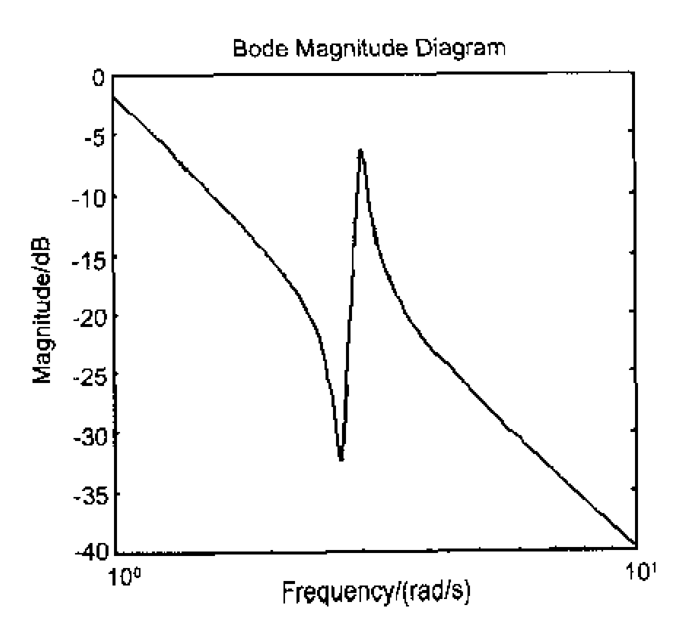


图 3-8 Bode 频率响应的振幅

4. evalfr

功能：计算系统单频率点的频率响应。

语法：frsp = evalfr(sys,f)

说明：evalfr 用于计算系统单个复数频率点的频率响应。其计算方法为：

$$H(f) = D + C(fI - A)^{-1}B$$

frsp = evalfr(sys,f)计算传递函数 LTI 对象 sys 在频率点 f 处的频率响应。其中，f 要以复数给出。

【例 3】 计算下述离散时间系统在 1+j 处的频率响应。

$$H(z) = \frac{z-1}{z^2+z+1}$$

```
H = tf([1 -1],[1 1 1],-1)
z = 1+j
evalfr(H,z)
```

MATLAB 返回：

```
ans =
```

2.3077e-01 + 1.5385e-01i

5. freqresp

功能：计算系统频率响应。

语法：H = freqresp(sys,w)

说明：freqresp 函数用于计算系统在实数频率处的频率响应。对于以状态空间描述的连续时间系统，其计算方法为：

$$H(j\omega) = D + C(j\omega I - A)^{-1}B$$

对于以传递函数描述的离散时间系统，首先对频率 w 进行以下转换：

$$z = e^{j\omega T}$$

其中，T_s 为采样周期，然后带入传递函数进行计算。

H = freqresp(sys,w)计算 LTI 对象 sys 在实数频率点 w 处的频率响应 H。其中 w 为频率点向量，H 为 3 维向量，其大小为

(number of outputs)×(number of inputs)×(length of w)

对于 SISO 系统，H(1,1,k)给出系统在频率 w(k)处的频率响应；对于 MIMO 系统，H(i,j,k)给处了第 I 个输入与第 j 个输出间在频率 w(k)处的频率响应。

【例 4】 计算下述系统在频率 1,10,100 处的频率响应。

$$P(s) = \begin{bmatrix} 0 & \dfrac{1}{s+1} \\ \dfrac{s-1}{s+2} & 1 \end{bmatrix}$$

```
P=[0 tf([0 1],[1 1]);tf([1 -1],[1 2]) 1]
w = [1 10 100]
H = freqresp(P,w)
```

MATLAB 返回：

```
H(:,:,1) =
     0              0.5000- 0.5000i
  -0.2000+ 0.6000i     1.0000
  H(:,:,2) =
     0              0.0099- 0.0990i
   0.9423+ 0.2885i     1.0000
H(:,:,3) =
     0              0.0001- 0.0100i
   0.9994+ 0.0300i     1.0000
```

6. interp

功能：在 FRD 模型频率点间插入频率响应数据。

语法：isys = interp(sys,freqs)

说明：isys = interp(sys,freqs)在 FRD 系统频率 freqs 处插入频率响应数据，freq 的单位必

须和 sys.frequency 的单位相同。频率 freqs 必须在系统的最小和最大频率之间（MATALB 不支持外插法）。返回值 isys 为包含新频率点处插值的数据的 frd 对象。

7. linspace

功能：给出相同间隔的行向量。

语法：y = linspace(a,b)

y = linspace(a,b,n)

说明：y = linspace(a,b)返回从 a 到 b 的 100 个相同间隔的数据组成的行向量；

y = linspace(a,b,n) 返回从 a 到 b 的 n 个相同间隔的数据组成的行向量。

8. logspace

功能：给出对数间隔的行向量。

语法：y = logspace(a,b)

y = logspace(a,b,n)

y = logspace(a,pi)

说明：y = logspace(a,b)返回从10^a到10^b的 50 个对数间隔的数据组成的行向量；

y = logspace(a,b,n) 返回从10^a到10^b的 n 个对数间隔的数据组成的行向量；

y = logspace(a,pi) 返回从10^a到 pi 的 50 个对数间隔的数据组成的行向量。

9. margin

功能：计算系统的增益和相角稳定裕度。

语法：[Gm,Pm,Wcg,Wcp] = margin(sys)

[Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)

margin(sys)

说明：margin 函数可从频率响应书记中计算处增益、相位裕度以及相应的交叉频率。增益和相位裕度示针对开环 SISO 系统而言的，它指示出当系统闭环时的相对稳定性。当不带输出变量引用时，margin 可在当前图形窗口中绘制出裕度的 Bode 图：

[Gm,Pm,Wcg,Wcp] = margin(sys)计算 LTI 对象 sys 的增益和相位裕度。返回值中，Gm 对应于系统的增益裕度，Wcg 为其响应的交叉频率；Pm 对应于系统的相位裕度，Wcp 为其响应的交叉频率。

[Gm,Pm,Wcg,Wcp} = margin(mag,phase,w)根据哟 mag、phase 和 w 给出的系统 bode 图数据计算系统的增益和相位裕度。其中 mag 给出 bode 图数据的幅值数据，phase 给出 bode 图数据的相位数据，w 为频率向量。

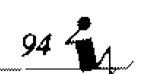
margin(sys)在当前图形窗口中绘制出系统裕度的 Bode 图。

【例 5】 计算下述离散时间系统的增援和相位裕度，并绘制出系统裕度的 Bode 图，如图 3-9 所示。

```
hd = tf([0.04798 0.0464],[1 -1.81 0.9048],0.1)
```

MATLAB 返回：

```
Transfer function:
 0.04798 z + 0.0464
---------------------
z^2 - 1.81 z + 0.9048
```

```
Sampling time: 0.1
[Gm,Pm,Wcg,Wcp] = margin(hd);
[Gm,Pm,Wcg,Wcp]
```

MATLAB 返回：

```
ans =
2.0517    13.5711    5.4374    4.3544
```

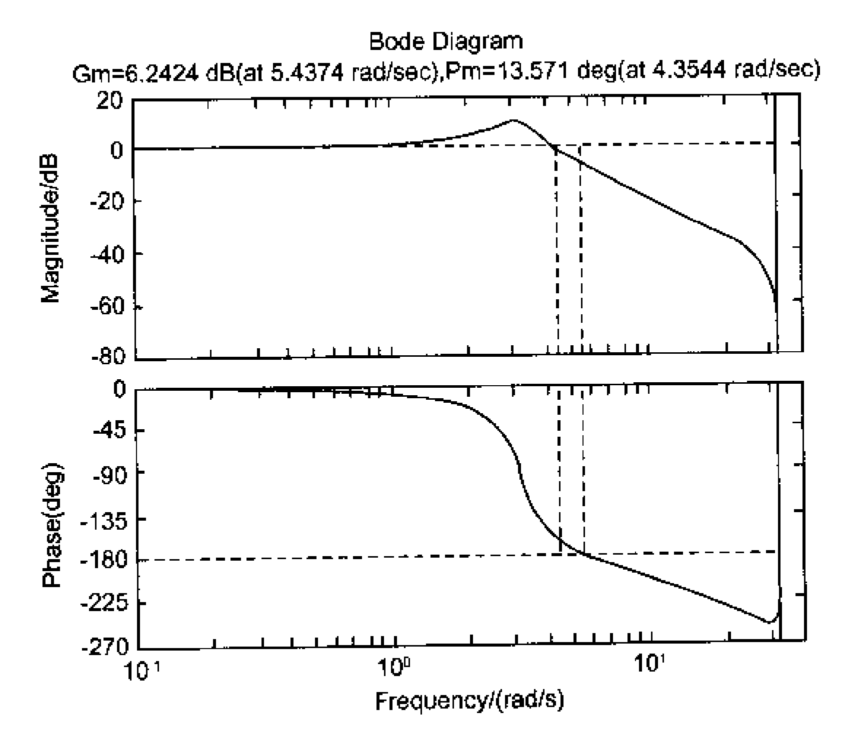


图 3-9　系统裕度的 Bode 图

10．ngrid

功能：Nichols 网格线绘制。

语法：ngrid

说明：ngrid 函数可给 Nichols 曲线图加上网格线，Nichols 曲线将 h/(1+h)与 h 相关联，当 h 为 SISO 系统的开环频率响应时，则 h/(1+h)为系统的单位闭环负反馈的频率响应。

Ngrid 绘制 Nichols 网格线，每条网格线具有同样的幅值和相位，幅值取-40dB ~ 40dB，相位取-360° ~0°。

【例 6】 绘制下述系统的 Nichols 曲线和网格线，如图 3-10 所示。

$$H(s)=\frac{-4s^4+48s^3-18s^2+250s+600}{s^4+30s^3+282s^2+525s+60}$$

```
H = tf([-4 48 -18 250 600],[1 30 282 525 60])
```

MATLAB 返回：

```
Transfer function:
- 4 s^4 + 48 s^3 - 18 s^2 + 250 s + 600
---------------------------------------
```

```
    s^4 + 30 s^3 + 282 s^2 + 525 s + 60
nichols(H)
ngrid
```

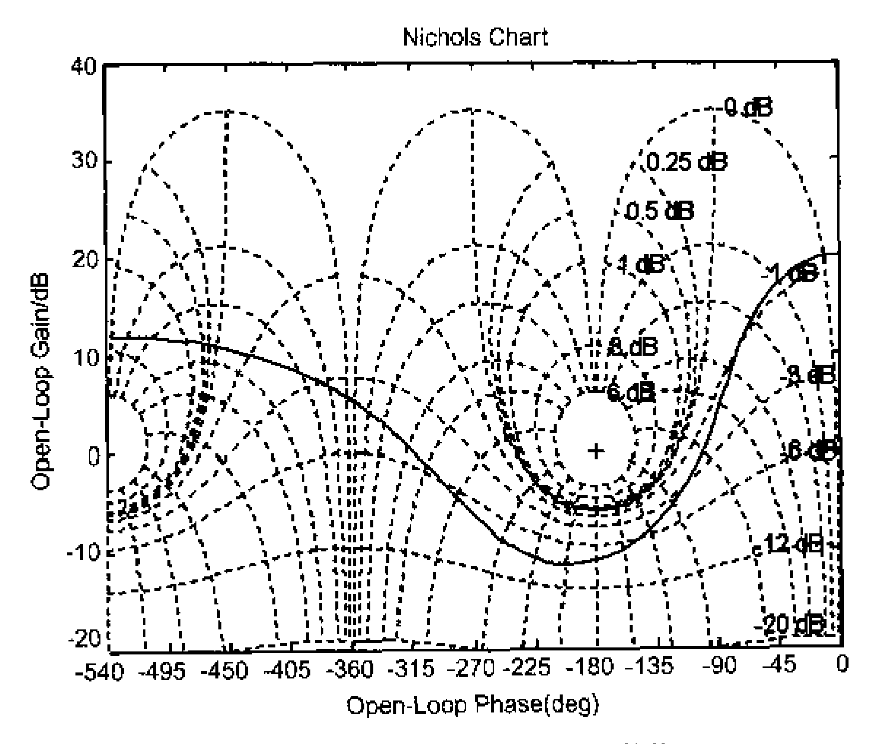


图 3-10　系统的 Nichols 曲线和网格线

11．nichols

功能：Nichols 绘制。

语法：nichols(sys)

nichols(sys,w)

nichols(sys1,sys2,...,sysN)

nichols(sys1,sys2,...,sysN,w)

nichols(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[mag,phase,w] = nichols(sys)

[mag,phase] = nichols(sys,w)

说明：nichols 函数可绘制 LTI 系统的 Nichols 频率响应曲线；

nichols(sys)：在当前窗口绘制系统 sys 的 Nichols 曲线。如果系统 sys 为 MIMO 系统，则该函数将产生一组 Nichols 曲线，每一条对应于一个输入输出通道。绘制的频率由系统零极点决定。

nichols(sys,w)：绘制在指定频率范围或频率点 w 的 Nichols 曲线。如果定义范围则 w=[wmin,wmax]；如果为频率点，则必须为频率向量。

nichols(sys1,sys2,...,sysN)：绘制多个系统的 Nichols 曲线。

nichols(sys1,sys2,...,sysN,w)：绘制多个系统在 w 处或范围的 Nichols 曲线。

nichols(sys1,'PlotStyle1',...,sysN,'PlotStyleN')：以指定的作图方式绘制不同系统的 Nichols 曲线。

[mag,phase,w] = nichols(sys)和[mag,phase] = nichols(sys,w)用于计算 Nichols 图的数据信息，并不作图。返回参数 mag 为 Nichols 图的幅度值，phase 为 Nichols 图的相位值，w 为 Nichols 图的频率点。mag 和 phase 均为三维向量，大小为

（输出变量数）×（输入变量数）×（w 的长度）

12. nyquist

功能：Nyquist 图绘制。

语法：

```
nyquist(sys)
nyquist(sys,w)
nyquist(sys1,sys2,...,sysN)
nyquist(sys1,sys2,...,sysN,w)
nyquist(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[re,im,w] = nyquist(sys)
[re,im] = nyquist(sys,w)
```

说明：nyquist(sys)绘制任意 LTI 系统 sys 的 Nyquist 响应，该系统可以为离散或连续的，SISO 或 MIMO 系统。当为 MIMO 系统时，函数将绘制一列 Nyquist 图。频率根据系统的零点和极点自动选择。

nyquist(sys,w)指明在频率 w= {wmin,wmax}范围内绘制 Nyquist 图。

nyquist(sys1,sys2,...,sysN)和 nyquist(sys1,sys2,...,sysN,w)绘制多个系统的 Nyquist 图。

[re,im,w] = nyquist(sys)和[re,im] = nyquist(sys,w)返回在频率 w 处的频率响应的实部和虚部，re 和 im 都是 3 维的阵列大小为

（输出变量数）×（输入变量数）×（w 的长度）

【例 7】 绘制如下系统的 Nyquist 图如图 3-11 所示。

$$H(s)=\frac{2s^2+5s+1}{s^2+2s+3}$$

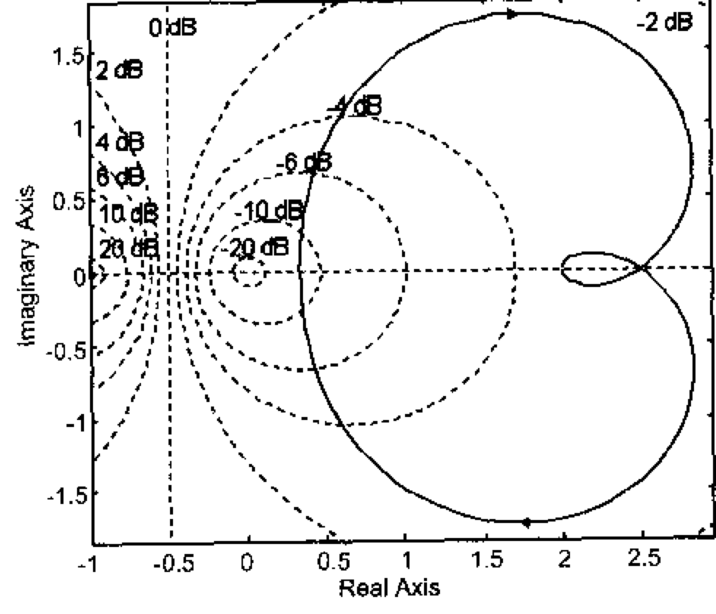


图 3-11　Nyquist 图

```
H = tf([2 5 1],[1 2 3]);
nyquist(H)
grid
```

13. sigma

功能：绘制 LTI 系统的奇异值 Bode 图。

语法：
```
sigma(sys)
sigma(sys,w)
sigma(sys,w,type)
sigma(sys1,sys2,...,sysN)
sigma(sys1,sys2,...,sysN,w)
sigma(sys1,sys2,...,sysN,w,type)
sigma(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[sv,w] = sigma(sys)
sv = sigma(sys,w)
```

说明：sigma 计算 LTI 系统频率响应的奇异值。对于 FRD 模型 sys.Response 系统计算在 sys.frequency 处 sys.Response 的奇异向量。

sigma(sys) 绘制任何 LTI 系统的奇异值图。

sigma(sys,w)在指定的范围 w = {wmin,wmax}内绘制图形。

sigma(sys,w,type)根据 type 选择作图，w 为可选参数：

- type=1：频率响应 H^{-1} 的奇异向量，H 为系统 sys 的频率响应；
- type=2：频率响应 I+H 的奇异向量；
- type=3：频率响应 $I+H^{-1}$ 的奇异向量。

这些参数只有在输入输出频道相同时使用。

sigma(sys1,sys2,...,sysN)、sigma(sys1,sys2,...,sysN,w)和 sigma(sys1,sys2,...,sysN,w,type)绘制多个系统的频率响应奇异值。

sigma(sys1,'PlotStyle1',...,sysN,'PlotStyleN')指定作图的颜色、线条等。

[sv,w] = sigma(sys)和 sv = sigma(sys,w)返回在频率 w 处的奇异值。当系统有 Nu 输入和 Ny 个输出时，sv 具有 min(Nu,Ny)行，sv 的列数和频率数（w 的长度）相同。

【例 8】 绘制如下系统 *H* 和 *I+H* 的奇异值，如图 3-12 所示。

$$H(s)=\begin{bmatrix} 0 & \dfrac{3s}{s^2+s+10} \\ \dfrac{s+1}{s+5} & \dfrac{2}{s+6} \end{bmatrix}$$

```
H = [0 tf([3 0],[1 1 10]) ; tf([1 1],[1 5]) tf(2,[1 6])]
subplot(211)
sigma(H)
grid
subplot(212)
sigma(H,[],2)
```

grid

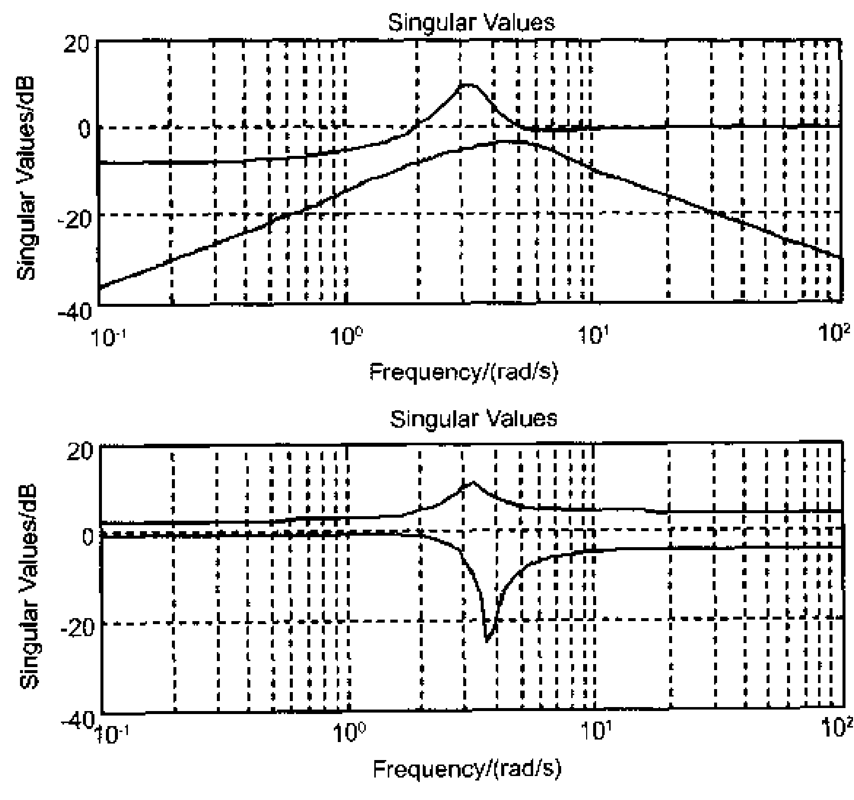


图 3-12　绘制 LTI 系统的奇异值 Bode 图

3.5 极点配置

关于极点配置函数见表 3-12。

表 3-12　极点配置函数列表

函数名	函数功能描述
acker	计算 SISO 极点配置
place	计算 MIMO 极点配置
estim	生成系统状态估计观测器
reg	系统调节器生成

1．acker

功能：计算 SISO 极点配置。

语法：k = acker(A,b,p)

说明：给定信号输入系统：

$$\dot{x} = Ax + bu$$

acker(A,b,p)利用 Ackermann 公式计算反馈增益矩阵 K，使采用全反馈 $u=-Kx$ 的单输入系统具有指定的极点 p。即：$p=eig(A-BK)$

acker 函数同样可以用来计算估计器增益（estimator gain），这时采用 l = acker(a',c',p).'形

式调用。

2．place

功能：计算 MIMO 极点配置。

语法：K = place(A,B,p)

[K,prec,message] = place(A,B,p)

说明：place(A,b,p)利用 Ackermann 公式计算反馈增益矩阵 K，使采用全反馈 $u=-Kx$ 的多输入系统具有指定的极点 p。即：$p=eig(A-BK)$

K = place(A,B,p)为多输入系统极点配置。

[K,prec,message] = place(A,B,p)同时返回反馈增益矩阵、系统闭环实际极点与期望极点 p 的接近程度 prec。prec 每个量为匹配的位数。如果实际极点与期望极点偏差 10%以上，则在 message 中给出警告信息。

【例 1】 考虑一个 2 输入、3 输出和 3 状态的系统（a,b,c,d），可以计算反馈增益矩阵，并将极点配置于：p = [1 1.23 5.0]。

```
p = [1 1.23 5.0];
K = place(a,b,p)
```

3．estim

功能：生成系统状态估计观测器。

语法：est = estim(sys,L)

est = estim(sys,L,sensors,known)

说明：est = estim(sys,L)利用系统状态空间模型 sys 和增益矩阵 L 生成系统的状态估计器。假定 sys 的所有输入为随机的，所有输出为可测的。对如下的连续时间系统：

$$\dot{x} = Ax + Bw$$
$$y = Cx + Dw$$

estim 函数将生成下列的张态和输入估计器：

$$\dot{\hat{x}} = A\hat{x} + L(y - C\hat{x})$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \hat{x}$$

对于离散时间系统，有类似的方程。

est = estim(sys,L,sensors,known)生成更加普遍的 LTI 系统 sys 的状态和估计器。sys 具有已知输入 u、随机输入 w 和可测输出 y，不可测输出 z。即对于如下连续时间系统：

$$\dot{x} = Ax + B_1 w + B_2 u$$

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} w + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} u$$

estim 函数将生成下列的状态和输出估计器：

$$\dot{\hat{x}} = A\hat{x} + B_2 u + L(y - C_2\hat{x} - D_{22}u)$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C_2 \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D_{22} \\ 0 \end{bmatrix} u$$

【例 2】 考虑一个有 4 输入、7 输出构成的状态空间模型。当系统使用输出的 4、7 和 1 作为传感器输入，输入 1、4 和 3 作为已知输入时，可以用 Kalman 增益 L 得到状态估计器。程序为

```
sensors = [4,7,1];
known = [1,4,3];
est = estim(sys,L,sensors,known)
```

4. reg

功能：系统调节器生成。

语法：rsys = reg(sys,K,L)

　　　rsys = reg(sys,K,L,sensors,known,controls)

说明：rsys = reg(sys,K,L)返回给定状态估计增益矩阵 L 即状态反馈增益矩阵 K 下的状态空间模型 sys 的调节器或补偿器 est，并假定系统的所有输出为可测的。

对于连续时间系统：

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

reg 生成如下的调节器，示意图如图 3-13 所示。

$$\dot{\hat{x}} = [A - LC - (B - LD)K]\hat{x} + Ly$$
$$u = -K\hat{x}$$

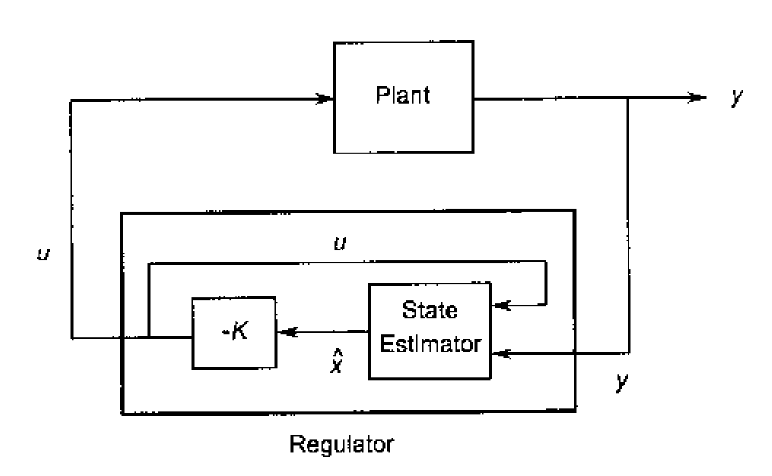


图 3-13　调节器示意框图

rsys = reg(sys,K,L,sensors,known,controls)生成更为普遍的 LTI 系统 sys 的状态和输出估计器。其中，sensors 用于指定可测输出，known 用于指定已知输入，controls 用于指定控制输入。这里假定系统同时具有已知输入 u_d，控制输入 u 和系统噪声 w，系统输出为非完全可测。生成的调节器如图 3-14 所示。

【例 3】 考虑一个有 4 输入、7 输出构成的状态空间模型。当系统使用输出的 4、7 和 1 作为传感器输入，输入 1、4 和 3 作为已知输入时，可以用 Kalman 增益 L 得到状态估计器。程序为

```
controls = [1,2,4];
sensors = [4,7,1];
known = [3];
regulator = reg(sys,K,L,sensors,known,controls)
```

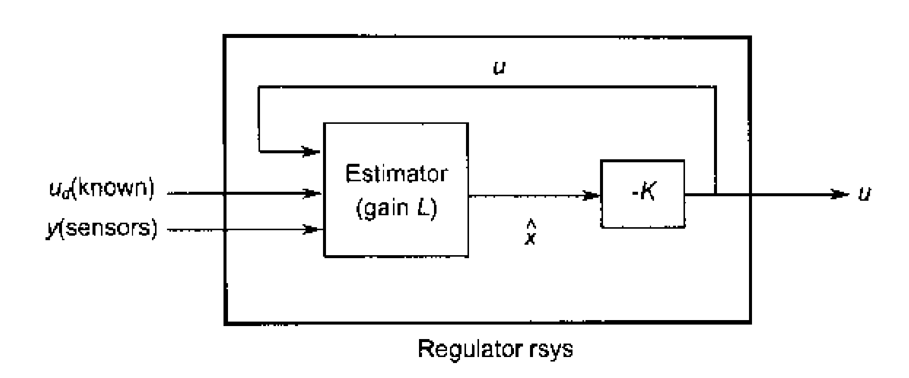


图 3-14 调节器示意框图

3.6 模型的综合处理

3.6.1 模型的转换

MATLAB 控制系统工具箱提供了模型间相互转换的函数，见表 3-13。

表 3-13 模型转换函数列表

函数名	函数功能描述
c2d	连续时间系统离散化
chgunits	转换 FRD 模型的 nunits 属性
d2c	离散时间系统连续化
d2d	离散时间系统重采样
reshape	转换 LTI 阵列的形状
residue	提供部分分式展开

1. c2d

功能：连续时间系统离散化。

语法：sysd = c2d(sys,Ts)
sysd = c2d(sys,Ts,method)
[sysd,G] = c2d(sys,Ts,method)

说明：sysd = c2d(sys,Ts)将连续时间 LTI 对象 sys 离散化，采样周期为 Ts，单位为秒。对输入采用零阶保持。

sysd = c2d(sys,Ts,method)定义离散化所采用的方法，其中 method 可以为以下字符串：

- 'zoh'：零阶保持，假设控制输入在采样周期内为常值；
- 'foh'：一阶保持，假设控制输入在采样周期内为线性；
- 'tustin'：采用双线性（tustin）逼近；
- 'prewarp'：采用改进的 tusitin 方法；
- 'matched'：采用 SISO 系统的零极点匹配法。

[sysd,G] = c2d(sys,Ts,method)返回矩阵 G，将连续初值条件 x_0 和 u_0 映射为对应的离散形式：

$$x[0] = G\begin{bmatrix} x_0 \\ u_0 \end{bmatrix}$$

$$u[0] = u_0$$

【例 1】 考虑系统：

$$H(s) = \frac{s-1}{s^2+4s+5}$$

输入时延 Td=0.35s，利用下面程序对系统进行离散化，采用一阶保持，采样周期为 Ts=0.1s。

```
H = tf([1 -1],[1 4 5],'inputdelay',0.35)
Hd = c2d(H,0.1,'foh')
```

MATLAB 输出为：

```
Transfer function:
                s - 1
exp(-0.35*s) * -------------
               s^2 + 4 s + 5
Transfer function:
        0.0115 z^3 + 0.0456 z^2 - 0.0562 z - 0.009104
z^(-3) * ---------------------------------------------
               z^3 - 1.629 z^2 + 0.6703 z
Sampling time: 0.1
```

对系统作图，如图 3-15 所示。

```
step(H,'-',Hd,'--')
```

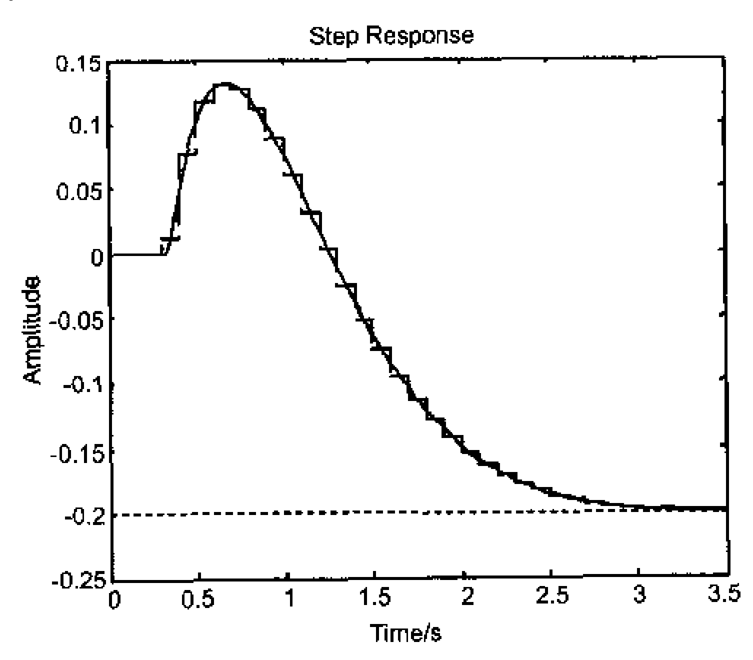


图 3-15　连续系统离散化

2．chgunits

功能：转换 FRD 模型的 nunits 属性。

语法：sys = chgunits(sys,units)

说明：sys = chgunits(sys,units)将储存在 FRD 对象 sys 中的频率单位转换为 units。units 为'Hz'或'rad/s'。

【例 2】

```
w = logspace(1,2,2);
sys = rss(3,1,1);
sys = frd(sys,w)
```

MATLAB 输出为：

```
From input 'input 1' to:
  Frequency(rad/s)      output 1
  ----------------     ------------------
       10        0.293773+0.001033i
      100        0.294404+0.000109i
Continuous-time frequency response data.
```

利用 chgunits 进行单位变换：

```
sys = chgunits(sys,'Hz')
sys.freq
```

MATLAB 输出为：

```
ans =
    1.5915
   15.9155
```

3. d2c

功能：离散时间系统连续化。

语法：sysc = d2c(sysd)
　　　sysc = d2c(sysd,method)

说明：sysc = d2c(sysd)离散时间 LTI 对象 sysd 的连续化。

sysc = d2c(sysd,method)定义连续化采用的方法。其中 method 可以为以下的字符串之一：

- 'zoh'：采用零阶保持，为缺省值；
- 'tustin'：采用双线性（tustin）逼近法；
- 'prewarp'：采用改进 tustin 方法；
- 'matched'：采用 SISO 系统的零极点匹配法。

【例 3】 考虑如下离散时间传递函数：

$$H(z)=\frac{z+0.2}{(z+0.5)(z^2+z+0.4)}$$

采样周期为：Ts=0.1。首先将系统连续化：

```
Ts = 0.1
H = zpk(-0.2,-0.5,1,Ts) * tf(1,[1 1 0.4],Ts)
Hc = d2c(H)
```

MATLAB 返回：



```
Warning: System order was increased to handle real negative poles.

Zero/pole/gain:
  -33.6556 (s-6.273) (s^2 + 28.29s + 1041)
 -----------------------------------------------------------
 (s^2 + 9.163s + 637.3) (s^2 + 13.86s + 1035)
```

再离散化：

```
c2d(Hc,Ts)
```

得到：

```
Zero/pole/gain:
      (z+0.5) (z+0.2)
------------------------------
(z+0.5)^2 (z^2 + z + 0.4)
Sampling time: 0.1
```

4. d2d

功能：离散时间系统重采样。

语法：sys1 = d2d(sys,Ts)

说明：sys1 = d2d(sys,Ts)将系统 sys 以 Ts 重新采样，该函数等价于：sys1 = c2d(d2c(sys),Ts)。

【例 4】 考虑传递函数为

$$H(z)=\frac{z-0.7}{z-0.5}$$

采用采样周期 Ts=0.05。

```
H = zpk(0.7,0.5,1,0.1)
H2 = d2d(H,0.05)
```

MATLAB 输出为：

```
Zero/pole/gain:
(z-0.8243)
----------
(z-0.7071)
Sampling time: 0.05
```

5. reshape

功能：转换 LTI 阵列的形状。

语法：sys = reshape(sys,s1,s2,...,sk)

　　　sys = reshape(sys,[s1 s2 ... sk])

说明：sys = reshape(sys,s1,s2,...,sk)或 sys = reshape(sys,[s1 s2 ... sk])将 LTI 阵列 sys 转换为 s1*s2*…*sk 维的 LTI 阵列。sys 系统必须具有 s1*s2*…*sk 个 LTI 对象。

【例 5】

```
sys = rss(4,1,1,2,3);
size(sys)
```

MATLAB 输出为：

```
2x3 array of state-space models
Each model has 1 output, 1 input, and 4 states.
```

使用 reshape 函数：

```
sys1 = reshape(sys,6);
size(sys1)
```

MATLAB 输出为

```
6x1 array of state-space models
Each model has 1 output, 1 input, and 4 states.
```

6. residue

功能：提供部分分式展开。

语法：[r,p,k] = residue(b,a)

[b,a] = residue(r,p,k)

说明：[r,p,k]=residue(b,a)返回分式多项式的极点留数表示，详见 MATLAB 数学函数说明。

3.6.2 模型的连接

MATLAB 中提供的模型连接函数见表 3-14。

表 3-14 模型连接函数列表

函数名	函数功能描述
append	多个 LTI 系统的组合
augstate	系统状态的输出增广
connect	框图建模
feedback	系统反馈连接
lft	两个 LTI 系统的 Redheffer 乘法
ord2	生成 2 阶模型
parallel	系统并联实现
series	系统串联实现
stack	将 LTI 模型放入到 LTI 对象阵列中

1. append

功能：多个 LTI 系统的组合。

语法：sys = append(sys1,sys2,...,sysN)

说明：append 函数实现对多个 LTI 系统的组合。对于 N 个子系统 sys1,sys2,...,sysN 组合过的系统如图 3-16 所示。

设各子系统的传递函数分别为 $H_1(s)$,...,$H_n(s)$，则合成后的子系统的传递函数为以下分块对角矩阵：

$$\begin{bmatrix} H_1(s) & 0 & \cdots & 0 \\ 0 & H_2(s) & & \vdots \\ \vdots & \cdots & \vdots & 0 \\ 0 & \cdots & 0 & H_N(s) \end{bmatrix}$$

对于已状态空间模型描述的两个子系统（A_1，B_1，C_1，D_1）和（A_2，B_2，C_2，D_2），append(sys1,sys2)组合的状态空间描述为

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

sys = append(sys1,sys2,...,sysN)多个 LTI 系统的组合。其中，各个子系统由 sys1,sys2,...,sysN 给出。

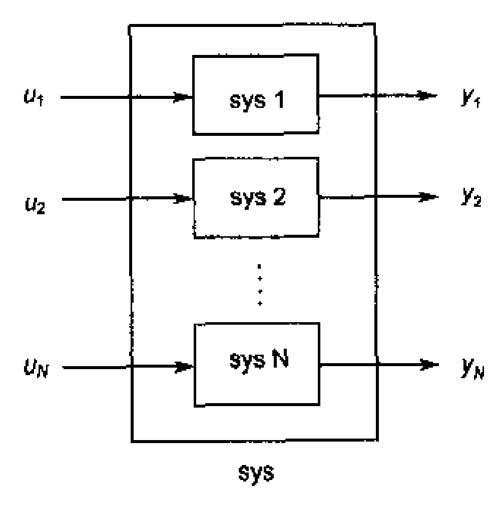


图 3-16　多系统组合框图

【例 1】

```
sys1 = tf(1,[1 0])
sys2 = ss(1,2,3,4)
sys = append(sys1,10,sys2)
```

MATLAB 返回：

```
a =
            x1        x2
    x1      0         0
    x2      0         1.00000
b =
            u1           u2       u3
    x1      1.00000      0        0
    x2      0            0        2.00000
c =
```

```
              x1          x2
      y1    1.00000       0
      y2      0           0
      y3      0         3.00000
d =
              u1      u2        u3
      y1      0       0         0
      y2      0    10.00000     0
      y3      0       0      4.00000
Continuous-time system.
```

2．augstate

功能：系统状态的输出增广。

语法：asys = augstate(sys)

说明：给定系统的以下状态空间模型：

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

则函数 augstate 将系统的状态向量增广至系统的输出中，从而形成系统：

$$\dot{x} = Ax + Bu$$
$$\begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} x + \begin{bmatrix} D \\ 0 \end{bmatrix} u$$

该函数常与函数 feedback 函数一起使用，构成全状态的反馈系统。

asys = augstate(sys)将系统的状态增广至系统的输出。其中，原系统由 sys 给出。

3．connect

功能：框图建模。

语法：sysc = connect(sys,Q,inputs,outputs)

说明：sysc = connect(sys,Q,inputs,outputs) 框图建模。其中，sys 通常为由函数 append 生成的无连接对角方块系统，见 append 函数。Q 矩阵用于指定系统 sys 的内部连接关系，inputs 和 outputs 用于选择系统 sysc 的输入和输出。

【例 2】 以图 3-17 所示的 SISO 传递函数系统模型构成的方框图为例，说明框图建模过程。

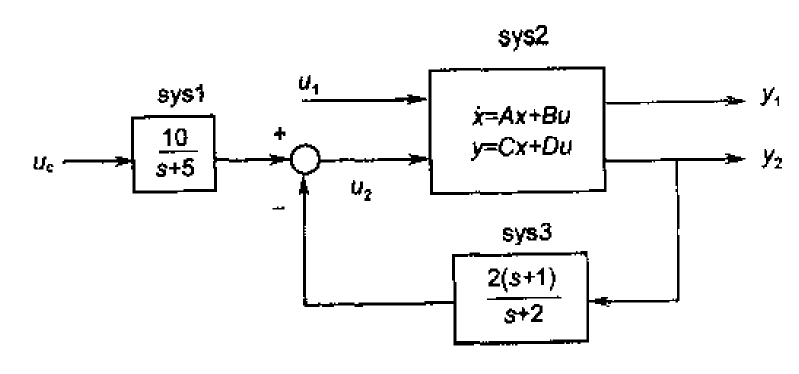


图 3-17 框图建模系统

(1) 定义传递函数或状态空间系统。

在方框图上对每个方框进行编号，并输入相应的传递函数分子分母系数：num1,num2, num3,⋯和 den1,den2,den3,⋯，或者输入状态矩阵(a,b,c,d)：a1,b1,c1,d1；a2,b2,c2,d2；a3,b3,c3,d3；⋯，或者定义零极点向量：z1,z2,z3,⋯和 p1,p2,p3,⋯及 k1,k2,k3,⋯；然后利用 ss，tf，zpk 函数对方框图建模。对于给出的例子，有

```
A = [ -9.0201   17.7791
      -1.6943   3.2138 ];
B = [ -0.5112   0.5362
      -0.002   -1.8470];
C = [ -3.2897   2.4544
      -13.5009   18.0745];
D = [-0.5476   -0.1410
    -0.6459   0.2958 ];
sys1 = tf(10,[1 5],'inputname','uc')
sys2 = ss(A,B,C,D,'inputname',{'u1' 'u2'},'outputname',{'y1' 'y2'})
sys3 = zpk(-1,-2,2)
```

(2) 建立无连接的状态空间模型。

通过调用 append 函数，形成一个由所有无连接的系统 sys。对于给出的例子，有

```
sys = append(sys1,sys2,sys3)
```

MATLAB 返回：

```
sys
a =
              x1        x2        x3        x4
       x1     -5        0         0         0
       x2      0     -9.0201    17.779      0
       x3      0     -1.6943    3.2138      0
       x4      0        0         0        -2
b =
              uc        u1        u2        ?
       x1      4        0         0         0
       x2      0     -0.5112    0.5362      0
       x3      0      -0.002    -1.847      0
       x4      0        0         0       1.4142
c =
              x1        x2        x3        x4
        ?     2.5       0         0         0
       y1      0     -3.2897    2.4544       0
       y2      0     -13.501    18.075       0
        ?      0        0         0      -1.4142
```

```
d =
           uc      u1        u2        ?
     ?     0       0         0         0
    y1     0      -0.5476   -0.141       0
    y2     0      -0.6459    0.2958      0
     ?     0       0         0         2
```

Continuous-time system.

(3) 指定方框间的连接关系。

矩阵 q 用于指示方框间的连接关系。矩阵的每一行对应于一个输入，其第一个元素为输入编号，其后为连接该输入的输出编号，如采用负连接，则以负值表示。对于上面的例子，有

```
Q = [3 1 -4
     4 3   0];
```

(4) 选择输入 /输出。

inputs 和 outputs 用于指示无连接系统中的某些输入/输出保留作为外部的输入/输出。对于上面的例子，有

```
inputs = [1 2];
outputs = [2 3];
```

(5) 内部连接。

通过调用 sysc = connect(sys,Q,inputs,outputs)这一函数，可以从矩阵 Q 中获得连接信息。对方框图进行连接，从而得到系统 sysc，其输入和输出分别由 inputs 和 outputs 确定。对于上面的例子，有

```
sysc = connect(sys,Q,inputs,outputs)
a =
           x1          x2          x3        x4
    x1    -5          0           0          0
    x2     0.84223    0.076636    5.6007     0.47644
    x3    -2.9012    -33.029     45.164     -1.6411
    x4     0.65708   -11.996     16.06      -1.6283
b =
           uc       u1
    x1     4        0
    x2     0       -0.076001
    x3     0       -1.5011
    x4     0       -0.57391
c =
           x1          x2         x3        x4
    y1    -0.22148    -5.6818    5.6568    -0.12529
    y2     0.46463    -8.4826   11.356      0.26283
```

d =

	uc	u1
y1	0	-0.66204
y2	0	-0.40582

Continuous-time system.

4．feedback

功能：系统反馈连接。

语法：sys = feedback(sys1,sys2)

sys = feedback(sys1,sys2,sign)

sys = feedback(sys1,sys2,feedin,feedout,sign)

说明：

sys = feedback(sys1,sys2)

sys = feedback(sys1,sys2,sign)

sys = feedback(sys1,sys2,feedin,feedout,sign)

5．lft

功能：两个 LTI 模型的 Redheffer 星乘法。

语法：sys = lft(sys1,sys2)

sys = lft(sys1,sys2,nu,ny)

说明：lft 计算两个 LTI 模型或 LTI 阵列的 Redheffer 星乘或 LFT。这种连接广泛应用于鲁棒控制中。

sys = lft(sys1,sys2,nu,ny)返回系统 sys1 和 sys2 的星乘，等价于图 3-18 的连接。

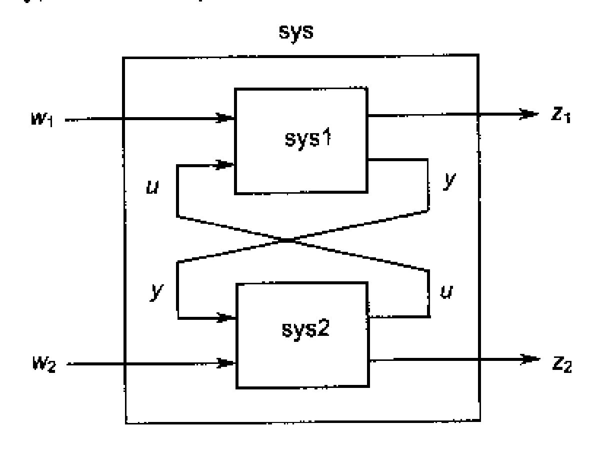


图 3-18　两个系统的 Redheffer 星乘连接

sys = lft(sys1,sys2)生成结果如图 3-19 所示。

- Lower LFT 连接：sys2 的输入输出频道数比 sys1 的小；
- Upper LFT 连接：sys1 的输入输出频道数比 sys2 的小。

6．ord2

功能：生成连续的二阶系统。

语法：[A,B,C,D] = ord2(wn,z)

[num,den] = ord2(wn,z)

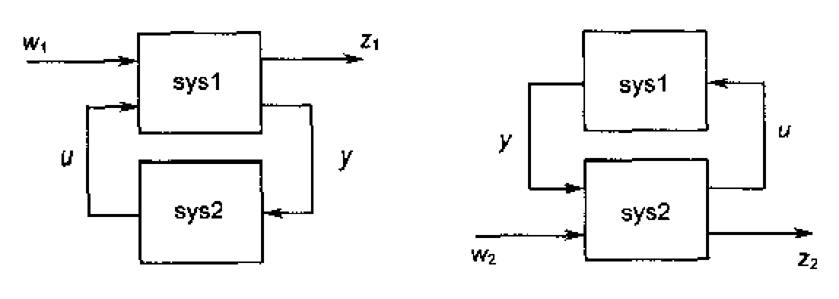


图 3-19 上、下两种 LFT 连接

说明：[A,B,C,D] = ord2(wn,z)生成二次的状态空间系统(A,B,C,D)：

$$h(s)=\frac{1}{s^2+2\zeta\omega_n s+\omega_n^2}$$

wn 为自然频率，z 为阻尼因子。

[num,den] = ord2(wn,z)返回二次传递函数模型的分子分母系数。

【例 3】 生成一个具有阻尼因子 0.4，自然频率 2.4rad/sec 的二阶转移函数，程序为

```
[num,den] = ord2(2.4,0.4)
sys = tf(num,den)
```

MATLAB 输出：

```
num =
      1
den =
    1.0000      1.9200      5.7600
Transfer function:
      1
--------------------
s^2 + 1.92 s + 5.76
```

7．parallel

功能：两个 LTI 模型的并联。

语法：sys = parallel(sys1,sys2)

sys = parallel(sys1,sys2,inp1,inp2,out1,out2)

说明：sys = parallel(sys1,sys2)返回 LTI 对象 sys1 和 sys2 的并联系统 sys。两个子系统必须都为连续时间系统或者同时为具有相同采样周期的离散时间系统。其并联形式如图 3-20 所示。

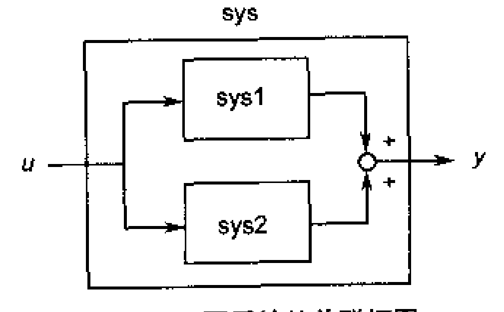


图 3-20 两系统的并联框图

sys = parallel(sys1,sys2,inp1,inp2,out1,out2)将 LTI 对象 sys1 和 sys2 按图 3-21 的方式连接：返回系统 sys 具有输入[v_1 , u , v_2]和输出[z_1 , y , z_2]。

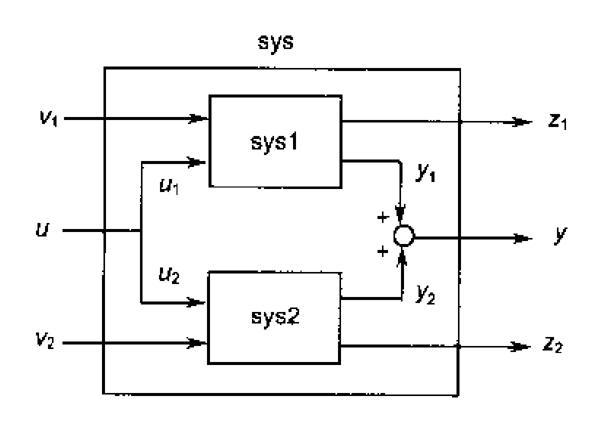


图 3-21　系统并联扩展框图

8. serise

功能：两个 LTI 系统的串联实现。

语法：sys = series(sys1,sys2)

sys = series(sys1,sys2,outputs1,inputs2)

说明：sys = series(sys1,sys2)返回 LTI 对象 sys1 和 sys2 的串联实现 sys。两个子系统必须都为连续时间系统或者同时为具有相同采样周期的离散时间系统。其并联形式如图 3-22 所示。

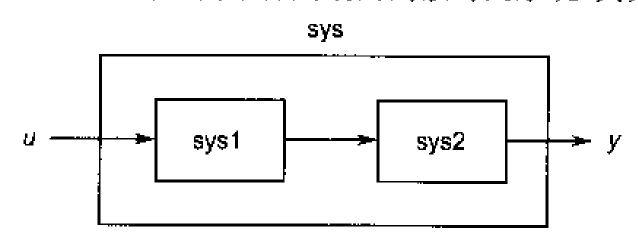


图 3-22　系统串联框图

sys = series(sys1,sys2,outputs1,inputs2)将 sys1 与 sys2 按图 3-23 连接。

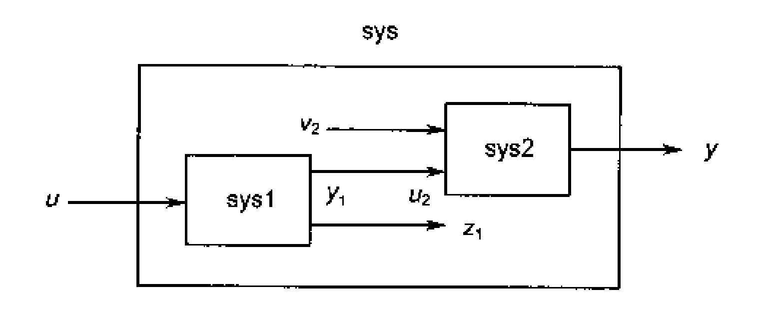


图 3-23　系统扩展串联框图

【例 4】 假设 sys1 具有 4 输入和 4 输出，sys2 具有 3 输入和 3 输出，将 sys1 的输出 2 和输出 4 串联到 sys2 的输入 1 和输入 2，采用如下的程序：

```
outputs1 = [2 4];
inputs2 = [1 2];
```

sys = series(sys1,sys2,outputs1,inputs2)

9．stack

功能：将 LTI 模型放入到 LTI 对象阵列中。

语法：sys = stack(arraydim,sys1,sys2,...)

说明：sys = stack(arraydim,sys1,sys2,...)生成一个维数为 arraydim 的 LTI 对象阵列，存放 sys1、sys2…。所有 LTI 对象必须具有相同的 I/O 维数。

【例 5】 假设 sys1 和 sys2 为具有相同 I/O 维数的 LTI 模型，则

- stack(1,sys1,sys2) 生成 2*1 的 LTI 阵列；
- stack(2,sys1,sys2) 生成 1*2 的 LTI 阵列；
- stack(3,sys1,sys2) 生成 1*1*2 的 LTI 阵列。

3.6.3 模型降阶

关于模型降阶的函数见表 3-15。

表 3-15 模型降阶函数

函数名	函数功能描述
balreal	状态空间的均衡实现
minreal	状态空间的最小实现
modred	模型降阶
sminreal	计算结构化模型降阶

1．balreal

功能：状态空间的均衡实现。

语法：sysb = balreal(sys)

[sysb,g,T,Ti] = balreal(sys)

说明：函数返回 LTI 系统 sys 的均衡实现。这种均衡模型通常用于模型的降阶。该函数可以用于离散或连续时间系统。

[sysb,g,T,Ti] = balreal(sys)同时返回均衡化参数 g，g 为 LTI 对象均衡实现相等的对角可控和可观的 gramians 矩阵。矩阵 T 为相似变换矩阵。

【例 1】

sys = zpk([-10 -20.01],[-5 -9.9 -20.1],1)

MATLAB 返回：

```
Zero/pole/gain:
    (s+10) (s+20.01)
----------------------
(s+5) (s+9.9) (s+20.1)
```

系统的均衡实现：

[sysb,g] = balreal(sys)

则 gramian 矩阵为：

```
g =
    0.1006
    0.0001
    0.0000
```

由 g 表明后面两个状态可以删除。

```
sysr = modred(sysb,[2 3],'del')
```

原系统的一阶近似为

```
zpk(sysr)
```

MATLAB 返回：

```
Zero/pole/gain:
  1.0001
--------
(s+4.97)
```

将原有系统与降阶后系统比较，如图 3-24 所示。

```
bode(sys,'-',sysr,'x')
```

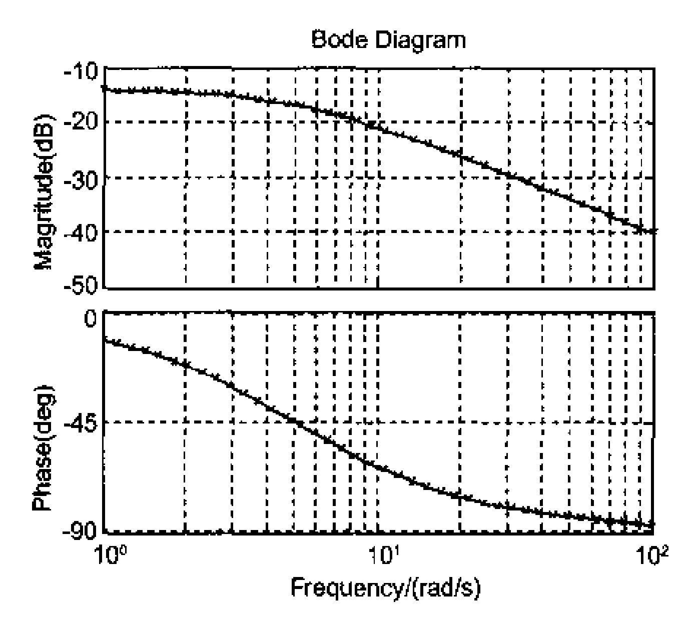


图 3-24　原系统与降阶后系统 Bode 图比较

2．minreal

功能：状态空间的最小实现。

语法：sysr = minreal(sys)

sysr = minreal(sys,tol)

[sysr,u] = minreal(sys,tol)

说明：sysr = minreal(sys)返回 LTI 对象 sys 的最小实现，即删除状态空间模型中的不可控和不可观状态；或者对消零点和极点对。

sysr = minreal(sys,tol)定义状态删除或零极点对消中允许的误差 tol。缺省为 tol=sqrt(esp)。

[sysr,u] = minreal(sys,tol)同时返回正交阵 U，使得(U*A*U',U*B,C*U')为(A,B,C)的 Kalman

分解。

【例 2】

```
g = zpk([],1,1)
h = tf([2 1],[1 0])
cloop = inv(1+g*h) * g
```

MATLAB 返回：

```
Zero/pole/gain:
          s (s-1)
 ------------------
 (s-1) (s^2 + s + 1)
```

对消 s=1 处的零极点

```
cloop = minreal(cloop)
```

MATLAB 返回：

```
Zero/pole/gain:
   s
 -------------
 (s^2 + s + 1)
```

3. modred

功能：模型降阶。

语法：rsys = modred(sys,elim)

rsys = modred(sys,elim,'mdc')

rsys = modred(sys,elim,'del')

说明：modred 用来降低系统的维数，通常与函数 balreal 连用。

rsys = modred(sys,elim)和 rsys = modred(sys,elim,'mdc')用于对 LTI 对象 sys 进降阶。索引向量 elim 给出需要删除的状态的索引。并且返回的降阶系统 rsys 具有和原系统向匹配的直流增益（DC gain）。

rsys = modred(sys,elim,'del')删除要删除的状态，调用不能保证系统具有与原系统相同的直流增益，单频域降阶后具有与原系统更好的近似。

【例 3】 考虑连续四阶模型：

$$H(s)=\frac{s^3+11s^2+36s+26}{s^4+14.6s^3+74.96s^2+153.7s+99.65}$$

要对该系统进行降阶，首先计算均衡状态空间实现：

```
h = tf([1 11 36 26],[1 14.6 74.96 153.7 99.65])
[hb,g] = balreal(h)
g'
```

MATLAB 返回：

```
ans =
    1.3938e-01    9.5482e-03    6.2712e-04    7.3245e-06
```

可以看出，最后 3 个元素很小，因此可以将其删除，同时使用两种方法，并进行比较 bode 图和阶跃响应图，如图 3-25 和图 3-26 所示。

```
hmdc = modred(hb,2:4,'mdc')
hdel = modred(hb,2:4,'del')
bode(h,'-',hmdc,'x',hdel,'*')
step(h,'-',hmdc,'-.',hdel,'--')
```

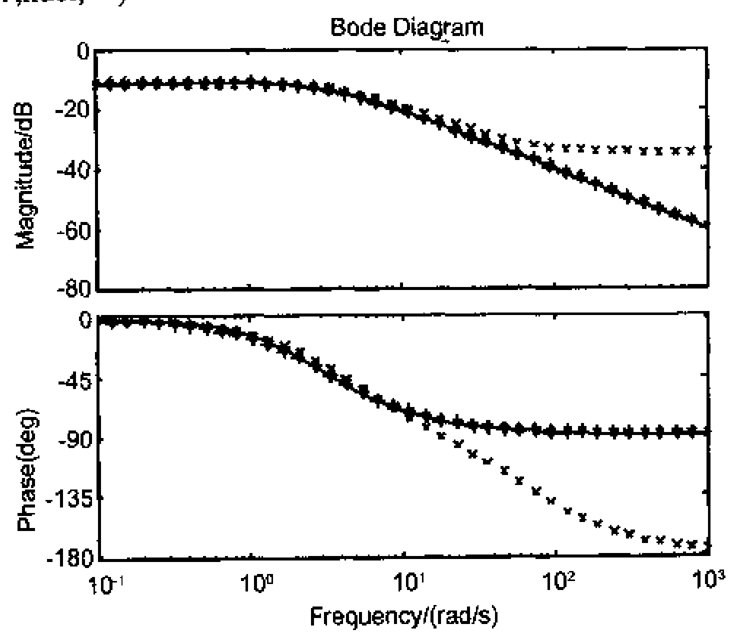


图 3-25　降阶系统与原系统 bode 图比较

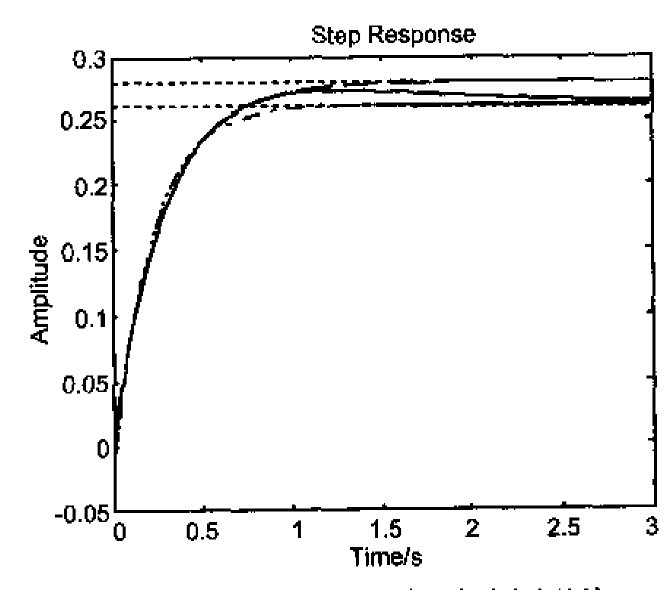


图 3-26　降阶后系统与原系统阶跃响应比较

4．sminreal

功能：计算结构化模型降阶。

语法：msys = sminreal(sys)

说明：msys = sminreal(sys)删除模型 sys 中对输入输响应无作用的状态。返回的状态空间模型 msys 为 sys 状态，并且输入输出响应和 sys 的相同。

【例 4】 考虑连接两个状态空间模型 sys1 和 sys2 如图 3-27 所示。

```
sys = [sys1,sys2];
```

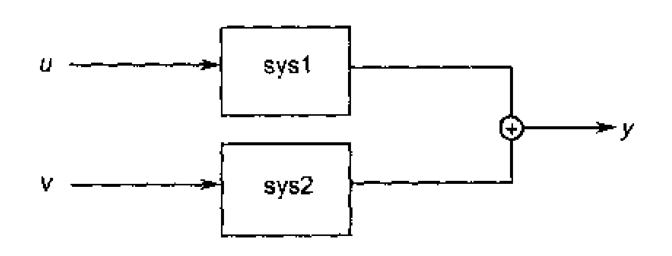


图 3-27 系统示意框图

如果通过 sys(1,1)获取 sys 中的子系统 sys1，则 sys，包括 sys2 的状态都保留了。要消除 sys2 中不可观的状态，保留 sys1 的状态，可以使用如下代码：

```
sminreal(sys(1,1))
```

3.7 LQG 设计

关于 LQG 设计函数，见表 3-16。

表 3-16 LQG 最优控制

函 数 名	函 数 功 能 描 述
lqr	连续系统的 LQ 调节器设计
dlqr	离散系统的 LQ 调节器设计
lqry	系统的 LQ 调节器设计
lqrd	计算连续时间系统的离散 LQ 调节器设计
kalman	系统的 kalman 滤波器设计
kalmd	连续系统的离散 kalman 滤波器设计
lqgreg	根据 kalman 估计器增益和状态反馈增益建立 LQG 调节器

1．lqr

功能：连续系统的 LQ 调节器设计。

语法：[K,S,e] = lqr(A,B,Q,R)

[K,S,e] = lqr(A,B,Q,R,N)

说明：[K,S,e] = lqr(A,B,Q,R,N)计算连续时间系统的最优反馈增益矩阵 K，使系统

$$\dot{x} = Ax + Bu$$

采用反馈律

$$u = -Kx$$

使性能指标函数

$$J(u) = \int_0^{\infty} (x^T Q x + u^T R u + 2x^T N u)\,\mathrm{d}t$$

最小。同时返回代数 Riccati 方程



$$A^TS+SA-(SB+N)R^{-1}(B^TS+N^T)+Q=0$$

的解 S 及闭环系统的特征值 *e*。缺省时 *N=0*。这里

$$K=R^{-1}(B^TS+N^T)$$

说明：该问题的数据必须

- （*A*，*B*）必须为稳定；
- *R*>0 并且

$$Q-NR^{-1}N^T\geqslant 0$$

- $Q-NR^{-1}N^T, A-BR^{-1}N^T$ 在虚轴上没有不可观测的模式。

2．dlqr

功能：离散系统的 LQ 调节器设计。

语法：[K,S,e] = dlqr(a,b,Q,R)
[K,S,e] = dlqr(a,b,Q,R,N)

说明：[K,S,e] = dlqr(a,b,Q,R,N) 计算离散时间系统的最优反馈增益矩阵 K，使系统

$$x[n+1]=Ax[n]+Bu[n]$$

采用反馈律

$$u[n]=-Kx[n]$$

使性能指标函数

$$J(u)=\sum_{n=1}^{\infty}(x[n]^TQx[n]+u[n]^TRu[n]+2x[n]^TNu[n])$$

最小。同时返回代数 Riccati 方程

$$A^TSA-S-(A^TSB+N)(B^TSB+R)^{-1}(B^TSA+N^T)+Q=0$$

的解 S 及闭环系统的特征值 *e*。缺省时 *N=0*。这里

$$K=(B^TSB+R)^{-1}(B^TSA+N^T)$$

说明：该问题的数据必须

- （*A*，*B*）必须为稳定；
- *R*>0 并且

$$Q-NR^{-1}N^T\geqslant 0$$

- $Q-NR^{-1}N^T, A-BR^{-1}N^T$ 在虚轴上没有不可观测的模式。

3．lqry

功能：系统的 LQ 调节器设计。

语法：[K,S,e] = lqry(sys,Q,R)
[K,S,e] = lqry(sys,Q,R,N)

说明：[K,S,e] = lqry(sys,Q,R,N)设计系统的最优反馈增益矩阵 K，使系统

$$\dot{x}=Ax+Bu$$
$$y=Cx+Du$$

采用反馈律

$$u=-Kx$$

使性能指标函数

$$J(u)=\int_0^\infty (y^T Qy+u^T Ru+2y^T Nu)\mathrm{d}t$$

最小。同时代数 Riccati 方程的解 S 及闭环系统的特征值 e。缺省时 $N=0$。该函数也可以计算相应的离散时间系统的最优反馈增益矩阵 K。

4. lqrd

功能：计算连续时间系统的离散 LQ 调节器设计。

语法：[Kd,S,e] = lqrd(A,B,Q,R,Ts)

[Kd,S,e] = lqrd(A,B,Q,R,N,Ts)

说明：[Kd,S,e] = lqrd(A,B,Q,R,N,Ts)设计连续系统的离散最优反馈增益矩阵 K，使系统

$$x[n+1]=Ax[n]+Bu[n]$$

采用反馈律

$$u[n]=-K_d x[n]$$

使性能指标函数

$$J=\int_0^\infty (x^T Qx+u^T Ru)\mathrm{d}t$$

最小。同时代数 Riccati 方程的解 S 及闭环系统的特征值 e。缺省时 $N=0$。

5. kalman

功能：系统的 kalman 滤波器设计。

语法：[kest,L,P] = kalman(sys,Qn,Rn,Nn)

[kest,L,P,M,Z] = kalman(sys,Qn,Rn,Nn)

[kest,L,P] = kalman(sys,Qn,Rn,Nn,sensors,known)

说明：kalman 函数用于对带有系统噪声和测量噪声的连续时间或者离散使就系统设计一个 kalman 最优状态估计器。对于下述连续时间系统

$$\dot{x}=Ax+Bu+Gw \qquad \text{(state equation)}$$

$$y_v=Cx+Du+Hw+v \qquad \text{(measurement equation)}$$

其中，w、v 满足

$$E(w)=E(v)=0,\quad E(ww^T)=Q$$

$$E(vv^T)=R,\quad E(wv^T)=N$$

则由 kalman 滤波器构成的状态估计器

$$x=A\hat{x}+Bu+L(y_v-C\hat{x}-Du)$$

$$\begin{bmatrix}\hat{y}\\ \hat{x}\end{bmatrix}=\begin{bmatrix}C\\ I\end{bmatrix}\hat{x}+\begin{bmatrix}D\\ 0\end{bmatrix}u$$

在最小化条件

$$P=\lim_{t\to\infty}E(\{x-\hat{x}\}\ \{x-\hat{x}\}^T)$$

下，为最优状态估计器。

这里增益矩阵 L 为下述代数 Riccati 方程的解，kalman 滤波器结构如图 3-28 所示。

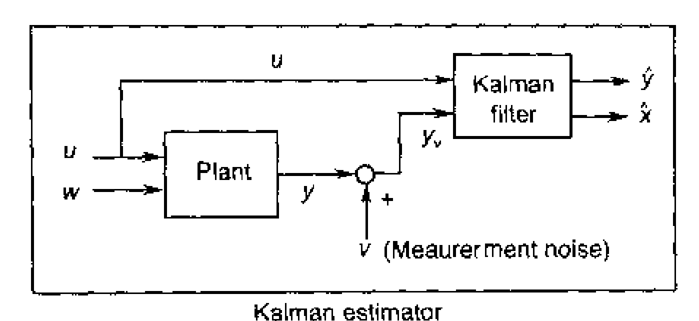


图 3-28　Kalman 估计器

对于离散时间系统：

$$x[n+1]=Ax[n]+Bu[n]+Gw[n]$$
$$y_v[n]=Cx[n]+Du[n]+Hw[n]+v[n]$$

其中，w、v 满足

$$E(w[n]w[n]^T)=Q,\quad E(v[n]v[n]^T)=R,$$
$$E(w[n]v[n]^T)=N$$

且 kalman 滤波器构成的状态估计器为

$$\hat{x}[n+1\mid n]=A\hat{x}[n\mid n-1]+Bu[n]$$
$$+L(y_v[n]-C\hat{x}[n\mid n-1]+Du[n])$$

$$\begin{bmatrix}\hat{y}[n\mid n]\\ \hat{x}[n\mid n]\end{bmatrix}=\begin{bmatrix}C(I-MC)\\ I-MC\end{bmatrix}\hat{x}[n\mid n-1]$$
$$+\begin{bmatrix}(I-CM)D & CM\\ -MD & M\end{bmatrix}\begin{bmatrix}u[n]\\ y_v[n]\end{bmatrix}$$

这里增益矩阵 L 和 M 由解离散 Riccati 方程得到。

[kest,L,P] = kalman(sys,Qn,Rn,Nn) 返回上述系统的 kalman 状态估计器 kest。其中 sys 为原系统的状态空间模型。Qn、Rn 和 Nn 上述的 Q、R 和 N 矩阵。返回值中 kest 为系统的状态估计器，L 和 P 对应于上述的 L 和 P 矩阵。

[kest,L,P,M,Z] = kalman(sys,Qn,Rn,Nn) 返回增益矩阵 L 和 M 和误差协方差矩阵：

$$P=\lim_{n\to\infty}E(e[n\mid n-1]e[n\mid n-1]^T),$$
$$e[n\mid n-1]=x[n]-x[n\mid n-1]$$
$$Z=\lim_{n\to\infty}E(e[n\mid n]e[n\mid n]^T),$$
$$e[n\mid n]=x[n]-x[n\mid n]$$

[kest,L,P] = kalman(sys,Qn,Rn,Nn,sensors,known) 生产更为普遍的 LTI 系统 sys 的 kalman 状态估计器。其中，sensors 用于指定可测输出，known 用于指定已知输入，其余输入为噪声输入。这里假定系统已知输入 u 和噪声输入 w 混在一起，系统输出为非完全可测。

6. kalmd

功能：连续系统的离散 kalman 滤波器设计。

语法：[kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)

说明：[kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)用于对带有系统噪声和测量噪声的连续时间系统设计一个离散的 kalman 最优状态估计器。

对于下述连续时间系统

$$\dot{x} = Ax + Bu + Gw \qquad \text{(state equation)}$$

$$y_v = Cx + Du + v \qquad \text{(measurement equation)}$$

其中，w、v 满足

$$E(w) = E(v) = 0, \quad E(ww^T) = Q_n,$$

$$E(vv^T) = R_n, \quad E(wv^T) = 0$$

kalmd 函数通过采用采样周期为 Ts 的零阶保持器离散化原连续时间系统，并通过将噪声协方差矩阵通过下式进行离散变换，将连续时间 Kalman 滤波器转换为等价的离散时间 Kalman 滤波器设计问题。然后计算该离散时间 Kalman 滤波器

$$Q_d = \int_0^{T_s} e^{A\tau} GQG^T e^{A^T\tau} d\tau$$

$$R_d = R / T_s$$

其中，sys 为原连续时间系统的状态空间模型。Qn 和 Rn 分别对应于上述的 Q 和 R 矩阵。Ts 为离散化时的采样周期。返回值中的 kest，L，P，M 和 Z 与 kalman 函数中的意义相同。

7．lqgreg

功能：根据 kalman 估计器增益和状态反馈增益建立 LQG 调节器。

语法：rlqg = lqgreg(kest,k)

rlqg = lqgreg(kest,k,'current')

rlqg = lqgreg(kest,k,controls)

说明：lqgreg 函数根据 kalman 估计器增益和状态反馈增益来建立 LQG 调节器。Kalman 估计器增益常由函数 kalman 生成，状态反馈增益常由 lqr、dlqr 及 lqry 生成，如图所示。对于连续时间系统，其调节器方程为

$$\dot{\hat{x}} = [A - LC - (B - LD)K]\hat{x} + Ly_v$$

$$u = -K\hat{x}$$

对于离散时间系统，其调节器方程可以基于状态估计或者一步预测。

$$u[n] = -K\hat{x}[n \mid n-1]$$

$$u[n] = -K\hat{x}[n \mid n]$$

其中，$\hat{x}$ 为 Kalman 滤波器的状态估计，y_v 为系统的测量输出，详见 kalman 函数。LQG 调节器如图 3-29 所示。

rlqg = lqgreg(kest,k)建立 LQG 调节器 rlqg，其中，kest 为 kalman 估计器增益，k 状态反馈增益。

rlqg = lqgreg(kest,k,'current')仅用于离散时间系统，定义调节器使用状态估计。'current'使用状态估计计算。缺省时则采用一步预测。

rlqg = lqgreg(kest,k,controls)定义其他系统输入，如图 3-30 中所示的 u_d。

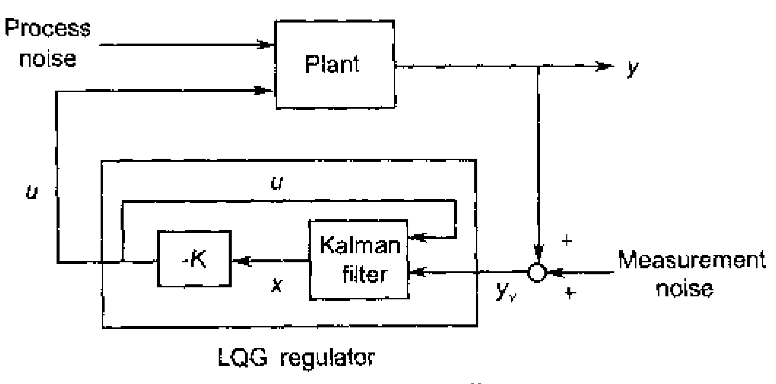


图 3-29　LQG 调节器

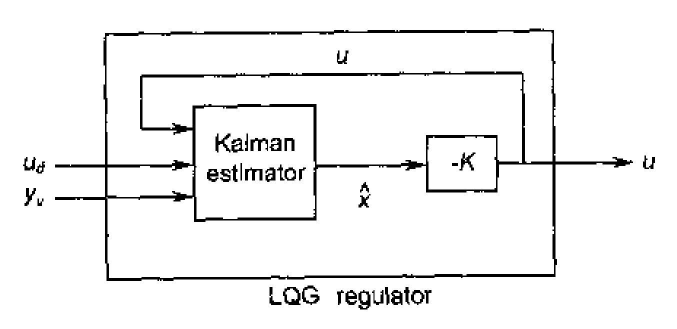


图 3-30　扩展 LQG 调节器

3.8　GUI 函数介绍

MATLAB 控制系统工具箱提供了可视化用户界面函数，见表 3-17。

图 3-17　GUI 函数列表

函　数　名	函 数 功 能 描 述
ltiview	激活 LTI 系统响应分析工具
sisotool	激活 SISO 设计工具

1．ltiview

功能：激活 LTI 系统响应分析工具。

语法：ltiview

```
ltiview(sys1,sys2,...,sysn)
ltiview('plottype',sys1,sys2,...,sysn)
ltiview('plottype',sys,extras)
ltiview('clear',viewers)
ltiview('current',sys1,sys2,...,sysn,viewers)
```

说明：ltiview 激活一个新的 LTI 观测器。

ltiview(sys1,sys2,...,sysn)激活一个包含 LTI 模型 sys1,sys2,...,sysn 阶跃响应的 LTI 观测器。

ltiview('plottype',sys1,sys2,...,sysn)定义要分析系统的响应类型 plottype。其中 plottype 可以为以下字符串之一：

- 'step'：系统阶跃响应；
- 'impulse'：系统脉冲响应；
- 'initial'：系统零输入响应；
- 'lsim'：系统的任意信号输入的时间响应；
- 'pzmap'：系统的零点极点图；
- 'bode'：系统的 Bode 图；
- 'nyquist'：系统的 Nyquist 图；
- 'nichols'：系统的 Nichols 图；
- 'sigma'：系统的奇异值响应。

ltiview('plottype',sys,extras)定义附加参数。要注意附加参数根据系统响应类型的不同而不同。

ltiview('clear',viewers) 通过句柄 viewers 清除 LTI 观测器的图像和数据。

ltiview('current',sys1,sys2,...,sysn,viewers)将系统 sys1,sys2,...,sysn 通过句柄 viewers 加入到 LTI 观测器中。如果新加入的系统和 LTI 观测器中原有的系统具有不同的 I/O 维数，则函数先清除 LTI 观测器中数据，这是只显示新的响应数据。

【例 1】 通过以下程序可以激活一个 LTI 观测器如图 3-31 所示，包含两个系统 sys1 和 sys2。

```
sys1 = rss(3,2,2);
sys2 = rss(4,2,2);
ltiview('step',sys1,sys2);
```

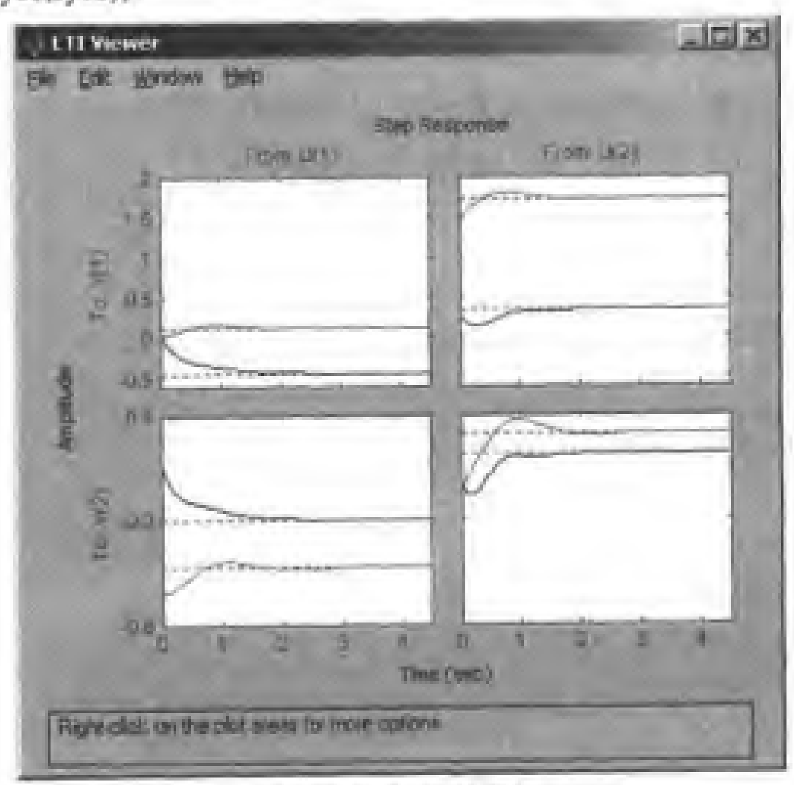


图 3-31 LTI 观测器

这里 LTI 观测器显示的为系统的阶跃响应图，要显示其他图可以通过菜单 Edit 下的 Plot Configuration 来配置显示的图的类型和顺序，配置窗口如图 3-32 所示。

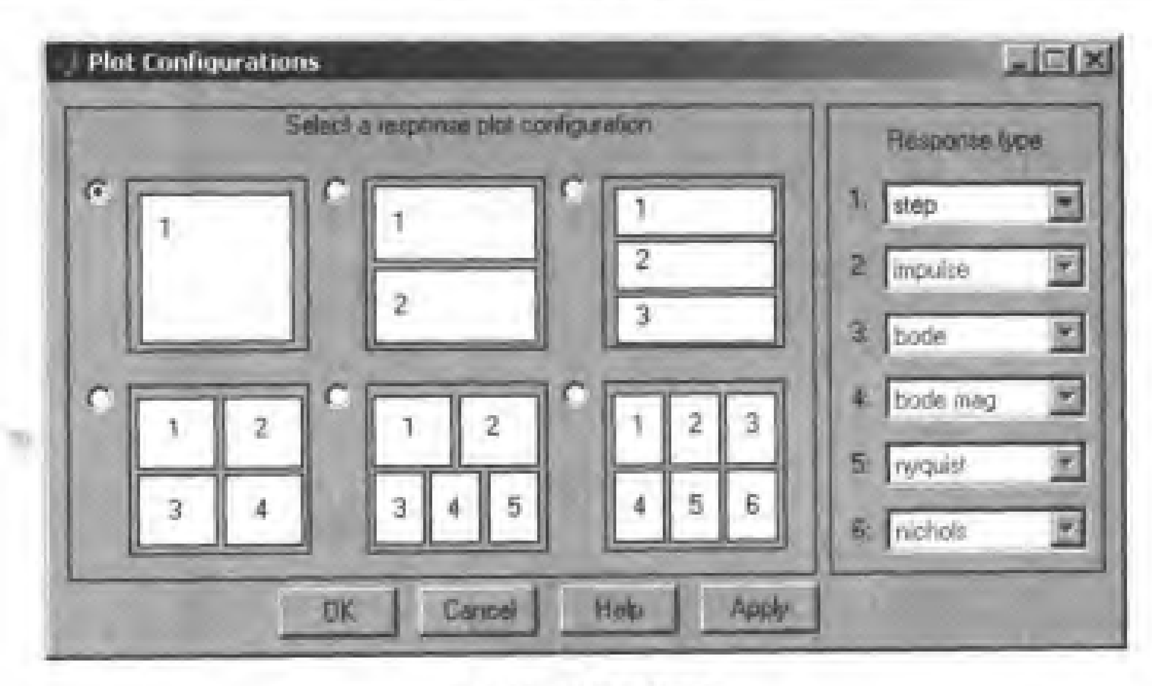


图 3-32　绘图配置窗口

2．sisotool

功能：激活 SISO 设计工具。

语法：sisotool

sisotool(plant)

sisotool(plant,comp)

sisotool(views)

sisotool(views,plant,comp,sensor,prefilt)

sisotool(views,plant,comp,options)

说明：当使用无输入参数调用 sisotool 时，函数激活一个 SISO 设计的图形用户界面（GUI）。输入参数 plant 为任意一个由 ts、ss 或 zpk 生成的 SISO LTI 模型。comp 为补偿器。sensor 为传感器。

views 可以为如下的字符串之一：

- 'rlocus'：根轨迹作图；
- 'bode'：开环响应的 Bode 图；
- 'nichols'：Nichols 作图；
- 'filter'：预过滤器 F 和闭环响应的 Bode 图。

【例 2】

```
H = [tf([1 1],[1 3 3 2])];
sys = ss(H);
sisotool(sys)
```

激活的 SISO 设计工具 GUI 如图 3-33 所示。

- 菜单条允许用户引入或者引出系统模型，并对其进行编辑。还可以对系统进行连续化或者离散化处理。

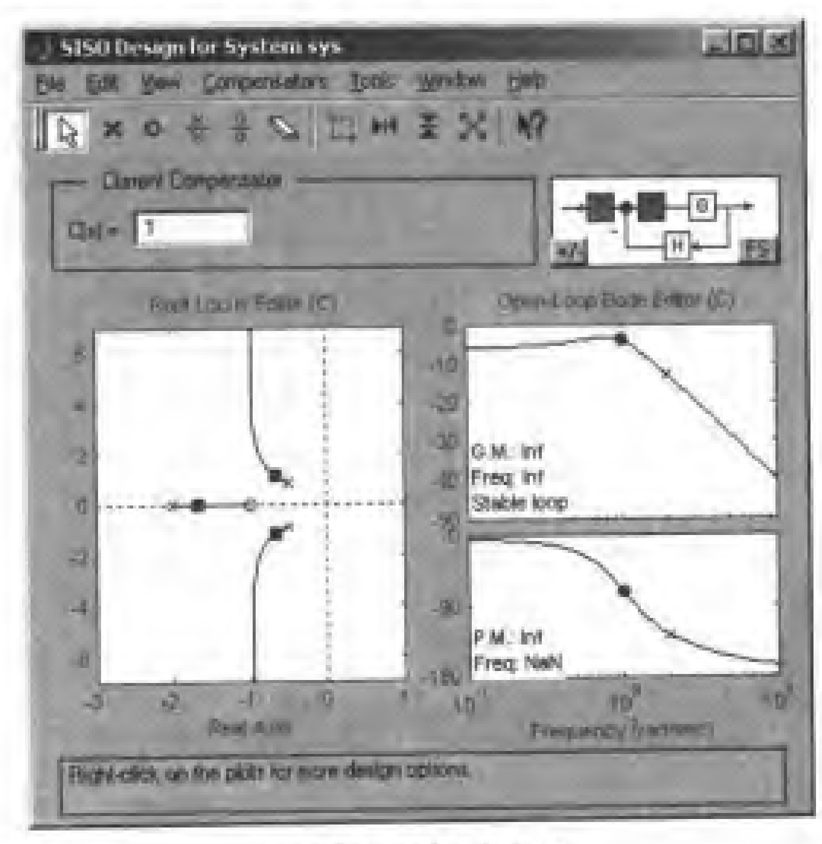


图 3-33 SISO 设计工具

- 反馈结构图解区给出当前反馈系统的整个结构。C 和 F 为补偿器，G 和 H 为系统的数据模块，用户可以根据需要使用这几个模块构造实际系统。单击各模块可以观察模块的当前属性。左下角的按钮允许用户在正负反馈间进行切换。
- 补偿器描述区给出当前补偿器的结构描述。
- 根轨迹工具条上的按钮允许用户增加或者删除补偿器的零极点，并可对闭环系统的极点进行拖曳。
- 绘制区用于显示系统的根轨迹和系统的波特图。

第4章 鲁棒控制工具箱

4.1 鲁棒控制理论及鲁棒控制工具箱简介

4.1.1 鲁棒控制理论概述

控制理论是研究被控对象状态的运动规律，其基础是反馈控制。反馈控制的主要目的是：

- 稳定被控对象；
- 改善被控对象的动态响应；
- 降低扰动对系统稳定性的影响，进而在系统稳定的前提条件下，降低扰动对系统性能的影响；
- 当被控对象不确定或变化时，仍能在稳定的状态下运行（稳定鲁棒性）；进一步，在系统稳定的前提条件下，当被控对象不确定或变化时，仍能在一定性能指标要求下运行（性能鲁棒性）。

20 世纪初，控制系统设计方法主要是基于 Bode 图和 Nyquist 图，利用间接的方法处理系统的不确定性问题，发展了在增益和相位存在变化时仍能保证闭环系统稳定的增益裕度和相位裕度概念。然而，遗憾的是这些处理方法大多局限于单变量输入单变量输出（SISO）系统。随着时间的推移，航天技术的发展要求处理大量的多变量输入多变量输出（MIMO）系统的设计问题，以二次型最优控制（LQ）为代表的一类多变量控制系统设计和最优化方法应运而生。但是随着其在实际工程中的应用，发现基于 LQ 理论设计出来的控制器对系统不确定性因素反应较为敏感，也就是说，不能保证闭环系统具有一定的稳定性和性能的鲁棒性，而且控制器设计过程要其准确知道干扰过程的全部统计特性，这一要求使该理论的工程应用受到工程实际条件的某些限制。另外，在实际工程应用过程中很难得到被控对象的精确数学模型，在控制系统设计过程中所采用的模型常常是在一定程度上经过近似化处理的数学模型。

为了克服上述已有的控制理论与方法所存在的共同缺陷，适应人类生产发展中对大量的不确定复杂对象的控制要求，鲁棒控制理论于 20 世纪 80 年代初开始形成和发展。目前已取得了大量的研究成果，成为控制学科的一个重要分支。鲁棒控制理论的研究目标是对存在广泛不确定性的对象提供有效的系统分析和控制器设计的方法，这里所说的广泛不确定性是指对不确定因素的统计特性不作任何假设，而仅认为它属于某个集合，相应的控制对象也因模型的不确定性构成对象族。

经过 20 多年的研究和发展，鲁棒控制理论取得了十分丰富的结果，如：内模控制理论，鲁棒调节器，稳定化控制器的 Youla 参数化，棱边定理，H_∞控制理论，结果奇异值理论和方

法，同时镇定理论和基于 Lyapunov 稳定性理论的系统鲁棒性分析和综合方法等。

4.1.2 鲁棒控制工具箱基本数据结构

MATLAB 鲁棒控制工具箱中的很多函数都是对一些基本的数据结构进行操作的，这些基本的数据结构见表 4-1。

表 4-1 鲁棒控制工具箱基本数据结构列表说明

函数名	函数功能说明
branch	从树型变量中返回某一分支元素
graft	添加一个根分支节点到一个树型变量中
issystem	判别系统 -变量
istree	识别树型变量
mksys	生成表示控制系统模型的变量
tree	生成数据变量
vrsys	返回标准系统变量名

1．branch

功能：从树型变量中返回某一分支元素。

语法：[b1,b2,,bn] = branch(tr,PATH1,PATH2,,PATHN)

说明：输入参数 tr 为树型变量，PATHi(i=1,2,…,n)为 tr 的 n 个分支路径名称：输出参数 bi(i=1,2,…,n)为 PATHi 对应的分支变量。分支路径名称为 -个如下形式的字符串：

PATH = '/name1/name2//namen';

其中 name1、name2、…、namen 为树型变量 tr 从根节点到某一分支节点的路径上各个节点的字符串名称。图 4-1 为树型的一个例子。

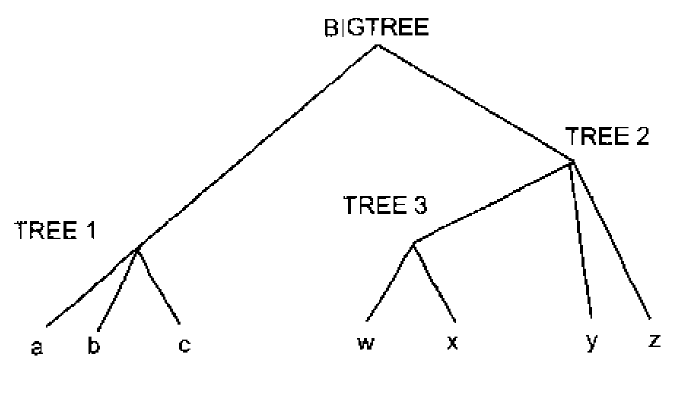


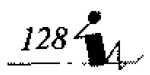
图 4-1 树型的例子

【例 1】

```
S = tree('a,b,c','foo',[49 50],'bar')
b1=branch(S,'c')
b1='bar'
```

2．graft

功能：添加一个根分支节点到一个树型变量中。



语法：TR = graft(TR1,B)

TR = graft(TR1,B,NM)

说明：输入参数 TR1 是一个树型变量，B 为任意类型的变量；输出参数 TR 为一个新的树型变量，该变量的根节点除包括 TR1 的所有根节点外，还包括添加的根节点 B。输入参数 NM 为字符串变量，用于指定新添加的根接点的名称。

【例 2】

```
S = tree('a,b,c','foo',[49 50],'bar')
TR=graft(S, 't', 'new')
b1=branch(TR, 'new')
b1='t'
```

3．issystem

功能：判断某一变量是否为通过 mksys 函数生成的树型系统模型变量。

语法：[I,TY,N] = issystem(S)

说明：输入参数 S 为一变量名；当 S 为通过 mksys 函数生成的树型系统模型变量时，输出参数 I=1，TY 为变量的模型类型，N 为分支变量个数；其他情况 I=0，TY 为空，n=0。

4．istree

功能：判断一个变量是否为树型变量。

语法：[i] = istree(T)

[i,b] = istree(T,path)

说明：当只有一个参数 T 时，函数用于判断一个变量是否为树型变量，输入参数 T 为待判断的变量，若 T 为树型变量则返回值 i=1，否则为 0；当指定参数 path 时，函数用于判断 T 是否在某一分支存在，该分支由参数 path 指定，若存在则为返回 i=1，参数 b 为指定的 T 的分支，否则返回 i=0。

5．mksys

功能：生成表示控制系统模型的变量。

语法：S = mksys(A,B,C,D)

S = mksys(v1,v2,v3,vn, TY)

说明：S = mksys(A,B,C,D)对应于系统模型的标准状态空间描述形式，即：

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

S = mksys(v1,v2,v3,vn, TY)具有可变的参数，对应多种系统模型描述，参数 TY 用于指定系统模型的形式，表 4-2 列出了不同的 TY 定义的系统模型和参数格式。

表 4-2 系统模型与参数格式

TY 值	参数 vi 的格式	系统模型
'ss'	(a,b,c,d,ty)	标准状态空间模型
'des'	(a,b,c,d,e,ty)	描述系统
'tss'	(a,b1,b2,c1,c2,d11,d12,d21, d22,e,ty)	两端口状态空间模型

（续）

TY 值	参数 vi 的格式	系统模型
'tdes'	(a,b1,b2,c1,c2,d11,d12,d21,d22,e,ty)	两端口描述子系统
'gss'	(sm,dimx,dimu,dimy,ty)	一般状态空间模型
'gdes'	(e,sm,dimx,dimu,dimy,ty)	一般描述系统
'gpsm'	(psm,deg,dimx,dimu,dimy,ty)	一般多项式系统矩阵
'tf'	(num,den,ty)	传递函数模型
'tfm'	(num,den,m,n,ty)	传递函数矩阵模型
'imp'	(y,ts,nu,ny)	脉冲响应模型

【例 3】 生成状态空间模型的系统变量：

```
a=[1,3;2,1];
b=[1 1];
c=[2,1;4,3];
d=[4,2];
s=mksys(a,b,c,d,'ss')
```

6．tree

功能：将多个向量和矩阵的数据集成到一个树型结构变量中。

语法：T = tree(nm,b1,b2,…,bn)

说明：tree 函数生成一个树型结构变量 T，包含所有分支 b1,b2,…,bn 的数据以及它们的名字（字符串变量）。参数 nm 为一个包含 n 个元素的向量，每个元素均为字符串变量，第 i 个元素作为 T 的分之 bi 的名称，nm 的各个元素用逗号隔开。

7．vrsys

功能：返回树型系统模型变量中具有指定模型类型的分支矩阵或向量变量名称。

语法：[VARS,N] = vrsys(NAM)

说明：输入参数 NAM 为具有如下形式的字符串变量：

[TY '_' SUF]

TY 为模型类型的名称，定义见表 4-2；字符串 SUF 为添加到返回变量名称的后缀；返回参数 VARS 为包含响应变量名称的字符串，各个变量名称用逗号隔开；N 为变量的个数。

【例 4】

```
[vars,n]=vrsys('ss_f')
```

MATLAB 返回：

```
vars =
        af,bf,cf,df
n =
      4
```




4.2　系统模型建立与转换工具

4.2.1　模型建立工具

在 MATLAB 鲁棒控制系统工具箱中提供了如表 4-3 中所列出的模型建立函数，使用这些函数可以建立复杂的控制系统模型。

表 4-3　模型建立函数列表说明

函数名	函数功能说明
augss, augtf	模型增广
Interc	建立一般的多变量互联系统模型

1．augss

功能：模型增广。

语法：[a,b1,b2,c1,c2,d11,d12,d21,d22] = augss(ag,bg,,aw1,bw1,,aw2,bw2,,aw3,bw3,)

[a,b1,b2,c1,c2,d11,d12,d21,d22] = augss(ag,bg,,aw1,bw1,,aw2,bw2,,aw3,bw3,,w3poly)

[tss] = augss(ssg,ssw1,ssw2,ssw3,w3poly)

[tss] = augss(ssg,ssw1,ssw2,ssw3)

说明：输入参数 ag、bg、cg 和 dg 为开环系统的状态空间矩阵；awi、bwi、cwi 和 dwi 分别为系统加权灵敏度传递函数矩阵对应的状态空间实现；

ssg、ssw1、ssw2 和 ssw3 分别为状态空间形式的系统模型变量。输出参数为双端口增广系统的状态空间矩阵或树型变量。

2．augtf

功能：建立具有灵敏度加权传递函数矩阵的增广系统模型。

语法：[a,b1,b2,c1,c2,d11,d12,d21,d22] = augtf(ag,bg,cg,dg,w1,w2,w3)

[tss] = augtf(ssg,w1,w2,w3)

说明：参数 ag，bg，cg，dg 为开环系统 *G* 的状态空间矩阵，ssg 为相应的系统模型变量，w1，w2 和 w3 为灵敏度加权传递函数矩阵；tss 为增广系统的状态空间矩阵或相应的树型系统变量 tss。

3．interc

功能：建立多变量互联系统的状态空间模型。

语法：[acl,bcl,ccl,dcl] = interc(a,b,c,d,m,n,f)

[sscl] = interc(ss,m,n,f)

说明：一个典型的多变量互联系统如图 4-2 所示。

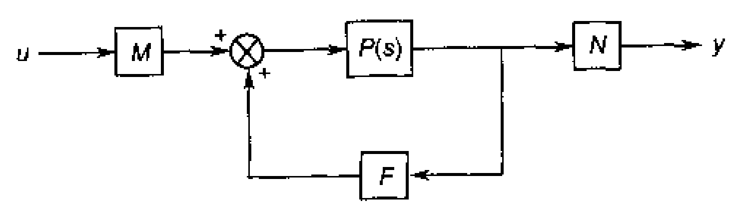


图 4-2　典型的多变量互联系统

闭环系统的状态空间的实现为：

A_{cl}，B_{cl}，C_{cl}，D_{cl}分别为：

$A_{cl} = A+BFXC$

$B_{cl} = B(M+FXDM)$

$C_{cl} = NXC$

$D_{cl} = NXDM$

其中 A，B，C，D 为 $P(s)$的状态空间实现。

函数输入参数 a,b,c,d 为图 4.2.1 中 $P(s)$对应的状态空间矩阵，m,n 和 f 为表示互联关系的常数矩阵。输出参数 acl,bc2,ccl,dcl 为闭环系统的状态空间矩阵，sscl 为树型状态空间模型变量。

【例 1】 考虑一个具有三个子系统（G_1，G_2，G_3）的系统，如图 4-3 所示。

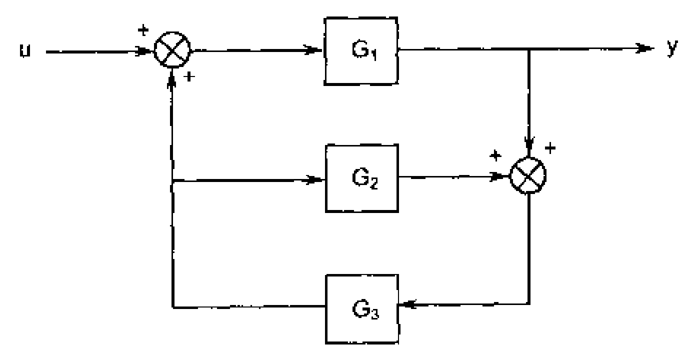


图 4-3 MIMO 互联系统

则有：

$$P(s)=\begin{pmatrix} G_1(s) & 0 & 0 \\ 0 & G_2(s) & 0 \\ 0 & 0 & G_3(s) \end{pmatrix}$$

且对于该问题，M，N 和 F 分别为

$$M=\begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

$$N=[I \quad 0 \quad 0]$$

$$F=\begin{bmatrix} 0 & 0 & I \\ 0 & 0 & I \\ I & I & 0 \end{bmatrix}$$

使用如下的 MATLAB 命令，可以得到 $P(s)$的状态空间实现：

```
[AA,BB,CC,DD] = append(A1,B1,C1,D1,A2,B2,C2,D2);
[AA,BB,CC,DD] = append(AA,BB,CC,DD,A3,B3,C3,D3);
```

4.2.2 模型转换工具

MATLAB 鲁棒控制工具箱提供了如表 4-4 所列出的模型间的转换函数。

表 4-4　模型转换函数列表说明

函数名	函数功能说明
bilin	频域多变量双线性函数
des2ss	将描述系统模型转换为状态空间模型
lftf	线型分式变换
sectf	扇区变换
stabproj	稳定/不稳定投影
slowfast	慢速/快速模态分解
tfm2ss	将传递函数矩阵转换为状态空间块控制器形式

1．bilin

功能：频域多变量双线性函数。

语法：[ab,bb,cb,db] = bilin(a,b,c,d,ver,Type,aug)

[ssb] = bilin(ss,ver,Type,aug)

说明：输入参数 a,b,c,d 为连续系统的状态空间矩阵，参数 ver 用于指定变换的方向，即有 s->z（正变换）或 z->s（逆变换），当 ver=1 时为正变换，ver= -1 时为逆变换。
参数 Type 用来指定变换的类型，aug 为对应不同变换类型的 Type 值，Type 为如下字符串之一：

(1) Type = 'Tustin'：Tustin 变换

$$s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

参数 aug=T，为采样周期；

(2) Type = 'P_Tust'

$$s=\frac{\omega_0}{\tan((\omega_0 T)/2)}\left(\frac{z-1}{z+1}\right)$$

参数 aug=[T　ω_0]；

(3) Type = 'BwdRec'

$$s=\frac{z-1}{Tz}$$

参数 aug=T；

(4) Type = 'FwdRec'

$$s=\frac{z-1}{T}$$

参数 aug=[T　h]；

(5) Type = 'S_Tust'

$$s=\frac{2}{T}\left(\frac{z-1}{\frac{z}{h}+1}\right)$$

参数 aug=[T h]；

(6) Type = 'S_ftjw'

$$s = \frac{z + p_1}{1 + z / p_2}$$

参数 aug=[p_2 p_1]；

(7) Type = 'G_Bilin'

$$s = \frac{\alpha z + \delta}{\gamma z + \beta}$$

参数 aug=[α β γ δ]。

【例 1】 考虑具有如下状态空间实现的连续时间系统（在 20Hz 处采样）：

$$A\begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

以下为四个从连续到离散的双线性变换，并绘制相应的波特图，结果如图 4-4 所示。

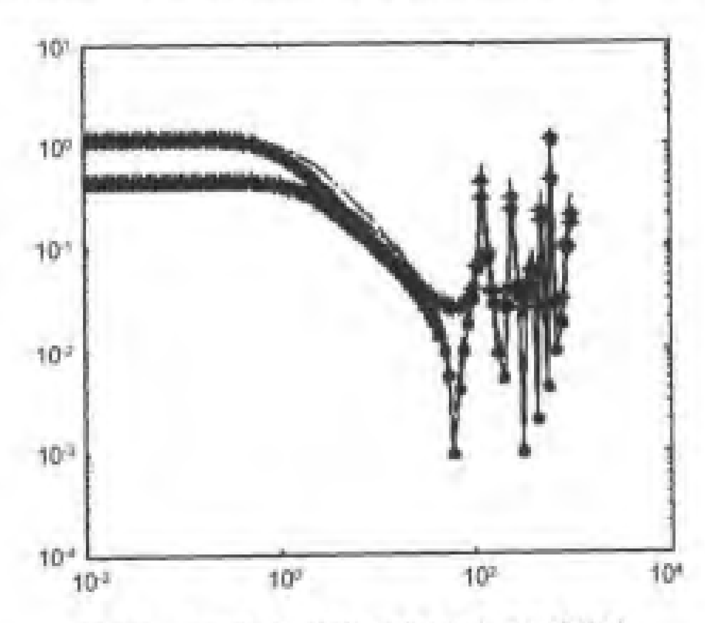


图 4-4　不同类型双线性变换离散奇异值波特图

程序为

```
a=[-1 1;0 -2];
b=[1 0;0 1];
c=[1 0;0 1];
d=[0 0;0 0];
ss = mksys(a,b,c,d);
[sst] = bilin(ss,1,'Tustin',0.05);
[ssp] = bilin(ss,1,'P_Tust',[0.05 40]);
[ssb] = bilin(ss,1,'BwdRec',0.05);
[ssf] = bilin(ss,1,'FwdRec',0.05);
```

```
w = logspace(-2,3,100);
svt = dsigma(sst,0.05,w);
svp = dsigma(ssp,0.05,w);
svb = dsigma(ssb,0.05,w);
svf = dsigma(ssf,0.05,w);
loglog(w,svt,'.-');
hold on;
loglog(w,svp,'--');
loglog(w,svb,'*');
loglog(w,svf,'-');
```

2. des2ss

功能：将描述系统模型转换为状态空间模型。

语法：[aa,bb,cc,dd] = des2ss(a,b,c,d,E,k)

　　　[ss1] = des2ss(ss,E,k)

说明：des2ss 通过对矩阵 E 使用奇异值分解(Singular Value Decomposition, SVD)：

$$E = U\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}V^T = [U_1\ \ U_2]\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V_1^T\\ V_2^T\end{bmatrix}$$

将描述系统：

$$G_1 := \left[\begin{array}{c|c} -Es + A & B \\ \hline C & D \end{array}\right]$$

转换为如下状态空间模型：

$$\left[\begin{array}{c|c} -Is + \Sigma^{-}(A_{11} - A_{12}A_{22}^{-1}A_{21})\Sigma^{-} & \Sigma^{-}(B_1 - A_{12}A_{22}^{-1}B_2) \\ \hline (C_1 - C_2A_{22}^{-1}A_{21})\Sigma^{-} & D - C_2A_{22}^{-1}B_2 \end{array}\right]$$

其中，矩阵 A_{ij} ，B_i，C_j 定义如下：

$$\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix} = \begin{bmatrix}U_1^T\\ U_2^T\end{bmatrix}A[V_1\ \ V_2]$$

$$\begin{bmatrix}B_1\\ B_2\end{bmatrix} = \begin{bmatrix}U_1^T\\ U_2^T\end{bmatrix}B$$

$$[C_1\ \ C_2] = C[V_1\ \ V_2]$$

函数输入参数 a,b,c,d,E 为描述系统模型的系数矩阵，ss 为有 mksys 建立的树型系统模型变量；k 为矩阵 E 的零空间的维数。

输出参数 aa,bb,cc,dd 为状态空间模型的系数矩阵，ss1 为对应的树型模型变量。

3. lftf

功能：单端口或双端口系统状态空间分式变换。

语法：[a,b1,b2,c1,c2,d11,d12,d21,d22] =

```
lftf(A,B1,B2,,a,b1,b2)
[aa,bb,cc,dd] =lftf(a,b1,b2,c1,c2,d11,d12,d21,d22,aw,bw,cw,dw)
[aa,bb,cc,dd] =lftf(aw,bw,cw,dw,a,b1,b2,c1,c2,d11,d12,d21,d22)
tss = lftf(tss1,tss2)
ss = lftf(tss1,ss2)
ss = lftf(ss1,tss2)
```

说明：给出如图 4-5 中系统：

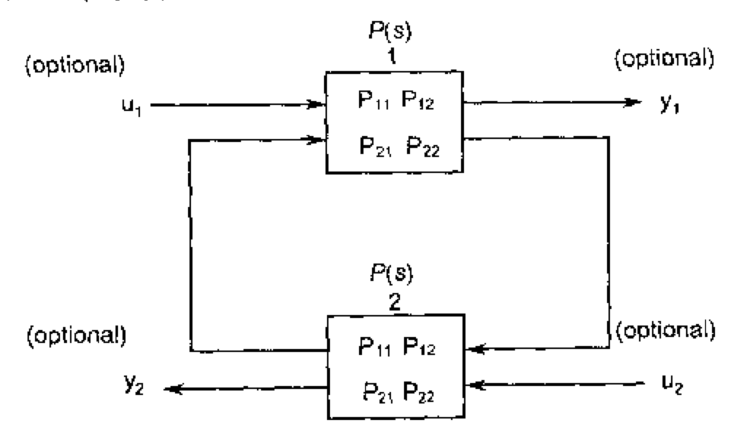


图 4-5 双端口线性分式变换

则函数 lftf 给出从输入 u1 到输出 y1 的传递函数：

$$tss1=\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

和从输入 u2 到输出 y2 的传递函数：

$$tss2=\left[\begin{array}{c|cc} a & b_1 & b_2 \\ \hline c_1 & d_{11} & d_{12} \\ d_2 & d_{21} & d_{22} \end{array}\right]$$

输入参数为双端口或单端口系统的状态空间矩阵对应的树型模型变量，输出参数为双端口或单端口闭环系统的状态空间矩阵 aa,bb,cc,dd 或对应的树型模型变量。

4．sectf

功能：状态空间扇区线性分式变换。

语法：

```
[ag,bg1,,dg22,at,bt1,,dt21,dt22] = sectf(af,bf1,,df22,secf,secg)
[ag,bg,cg,dg,at,bt1,,dt21,dt22] = sectf(af,bf,cf,df,secf,secg)
[tssg,tsst] = sectf(tssf,secf,secg)
[ssg,tsst] = sectf(ssf,secf,secg)
```

说明：输出参数 a,bfi,cfi,dfij（i=1,2）为双端口系统 F 的状态空间矩阵；

af,bf,cf,df 为单端口系统的状态空间矩阵；

tssf,ssf 为系统的树型模型变量；

secf,secg 分别为系统 F 和 G 的扇区性能指标，它们的取值与对应的性能指标见表 4-5。

表中 A，B 为常数或方阵，a，b 为常数向量；S 为分块矩阵 $S=[S_{11}\ S_{12};\ S_{21}\ S_{22}]$其分块可以为数或方阵，tsss 为双端口系统树型模型变量，tsss=mksys(a,b1,b2,,'tss')，其传递函数为

$$S(s)=\begin{bmatrix} S_{11}(s) & S_{12}(s) \\ S_{21}(s) & S_{22}(s) \end{bmatrix}$$

表 4-5　扇区性能指标表

secf 或 secg 的值	扇区性能指标
[-1,1]或[-1;1]	$\|y\|^2 \leqslant \|u\|^2$
[0,Inf]或[0;Inf]	$\mathrm{Re}[y*u] \geqslant 0$
[A,B]或[A;B]	$\mathrm{Re}[(y-Au)*(y-Bu)] \leqslant 0$
[a,b]或[a;b]	$\mathrm{Re}[(y-\mathrm{diag}(a)u)*(y-\mathrm{diag}(b)u)] \leqslant 0$
S	$\mathrm{Re}[(S_{12}y+S_{11}u)*(S_{22}y+S_{21}u)] \leqslant 0$
tsss	$\mathrm{Re}[(S_{12}y+S_{11}u)*(S_{22}y+S_{21}u)] \leqslant 0$

输出参数 ag,bgi,cgi,dgij（i,j=1,2）为双端口系统 F 的状态空间矩阵；

ag,bg,cg,dg 为单端口系统 F 的状态空间矩阵；

at,bti,cti,dtij（i,j=1,2）为系统 F 和 G 的线性分式变换对应的状态空间矩阵；

【例 2】 有如下扇区变换：

$$P(s)=\frac{1}{s+1}\in \mathrm{sector}[-1,1]\to P_1(s)=\frac{s+2}{s}\in \mathrm{sector}[0,\infty]$$

可以通过如下代码实现，并绘制 Nyquist 曲线如图 4-6 和图 4-7 所示。

```
[A,B,C,D] = tf2ss(1,[1 1]);
[a,b,c,d] = sectf(A,B,C,D,[-1,1],[0,Inf]);
nyquist(A,B,C,D)
 nyquist(a,b,c,d)
```

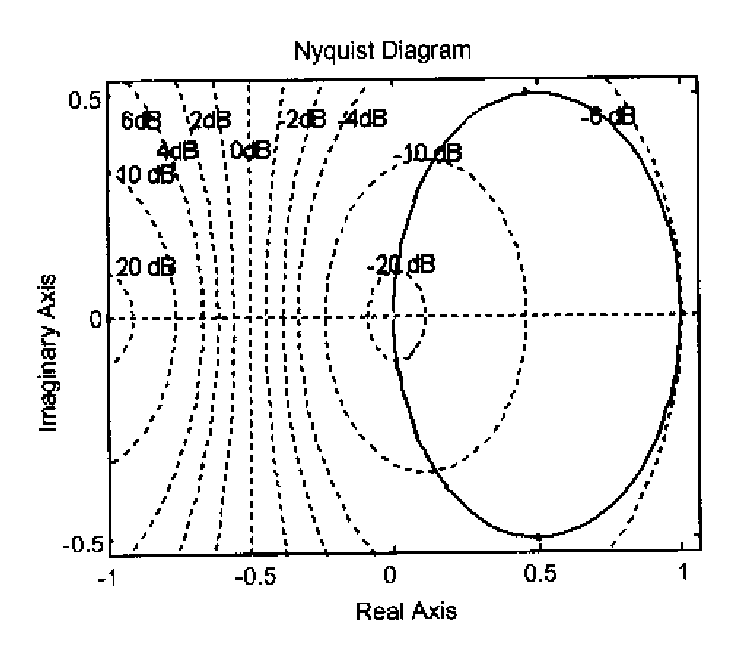


图 4-6　系统 $P(s)=1/(1+s)$的 Nyquist 图

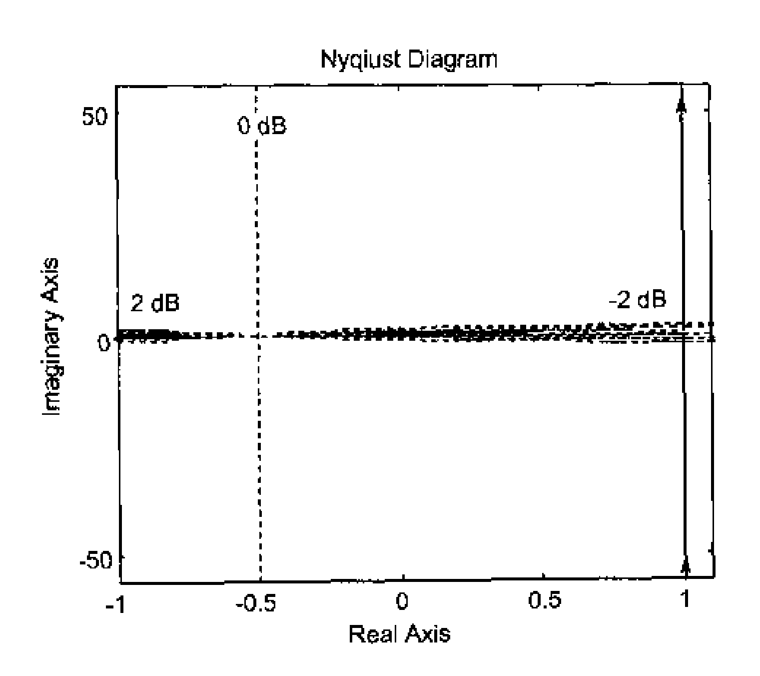


图 4-7 系统 $P_1(s)=(s+2)/s$ 的 Nyquist 图

5. stabproj

功能：状态空间的稳定和不稳定投影。

语法：[a1,b1,c1,d1,a2,b2,c2,d2,m] = stabproj(a,b,c,d)

[ss1,ss2,m] = stabproj(ss)

说明：stabproj 计算最小系统 $G(s)$的稳定和不稳定投影，使得：

$$G(s)=[G(s)]_{-}+[G(s)]_{+}$$

其中

$$[G(s)]_{-}:=(\hat{A}_{11},\ \hat{B}_1,\ \hat{C}_1,\ \hat{D}_1)$$

为系统稳定部分；

$$[G(s)]_{+}:=(\hat{A}_{22},\ \hat{B}_2,\ \hat{C}_2,\ \hat{D}_2)$$

为系统不稳定部分；

输入参数 a,b,c,d 为系统状态空间矩阵；ss 为系统模型对应的树型模型变量。

输出参数 a1,b1,c1,d1 为系统稳定状态空间投影，ss1 为对应的树型模型变量；a2,b2,c2,d2 为系统稳定状态空间投影，ss2 为对应的树型模型变量；

m 为系统矩阵 A 的稳定特征值的个数。

6. slowfast

功能：状态空间的慢速和快速模态分解。

语法：[a1,b1,c1,d1,a2,b2,c2,d2] = slowfast(a,b,c,d,cut)

[ss1,ss2] = slowfast(ss,cut)

说明：slowfast 计算系统 $G(s)$的慢速和快速模态分解，使得：

$$G(s)=[G(s)]_s+[G(s)]_f$$

其中

$$[G(s)]_s:=(\widehat{A}_{11},\ \widehat{B}_1,\ \widehat{C}_1,\ \widehat{D}_1)$$

为慢速模态部分

$$[G(s)]_f:=(\widehat{A}_{22},\ \widehat{B}_2,\ \widehat{C}_2,\ \widehat{D}_2)$$

为快速模态部分。

输入参数 a,b,c,d 为系统状态空间矩阵；ss 为系统模型对应的树型模型变量；cut 为模态分解指数。

输出参数 a1,b1,c1,d1 为系统慢速模态，ss1 为对应的树型模型变量；a2,b2,c2,d2 为系统快速模态，ss2 为对应的树型模型变量。

关于 stabproj 和 slowfast 的算法说明如下：

对于状态空间模型利用函数 blksch 或 rschur 对矩阵 A 进行 Shur 分解，使得：

$$A=V^TAV=\begin{bmatrix}\widehat{A}_{11} & \widehat{A}_{12}\\ 0 & \widehat{A}_{22}\end{bmatrix}$$

当采用不同的有序 Schur 分解形式时，分别得到对应于稳定和不稳定投影以及对应于慢速快速模态分解的$\widehat{A}_{11}\widehat{A}_{22}$和，其中后者满足：

$$\left|\lambda_i(\widehat{A}_{11})\right|<\left|\lambda_i(\widehat{A}_{22})\right|$$

然后求解下面的矩阵方程以计算矩阵 X：

$$\widehat{A}_{11}X-X\widehat{A}_{22}+\widehat{A}_{12}=0$$

最后得到状态空间投影：

$$[G(s)]_-\quad or\quad [G(s)]_s:=\left[\begin{array}{c|c}\widehat{A}_{11} & \widehat{B}_1\\ \hline C_1 & 0\end{array}\right]$$

$$[G(s)]_+\quad or\quad [G(s)]_f:=\left[\begin{array}{c|c}\widehat{A}_{22} & \widehat{B}_2\\ \hline C_2 & D\end{array}\right]$$

其中

$$\begin{bmatrix}\widehat{B}_1\\ \widehat{B}_2\end{bmatrix}:=\begin{bmatrix}I & -X\\ 0 & I\end{bmatrix}$$

$$\left[\widehat{C}_1\ \middle|\ \widehat{C}_2\right]:\ =CV^T\begin{bmatrix}I & X\\ 0 & I\end{bmatrix}$$

7. tfm2ss

功能：将传递函数矩阵 $G(s)$转换为状态空间块控制器形式。

语法：[a,b,c,d] = tfm2ss(num,den,r,c)

[ss] = tfm2ss(tf,r,c)

说明：函数 tfm2ss 用于将传递函数矩阵 $G(s)$转换为状态空间控制器的形式，其中传递函

数矩阵 $G(s)$具有如下的形式：

$$G(s)=\frac{1}{d(s)}N(s)_{r\times c}$$

上式中，$N(s)$为 r*c 矩阵，其第 i 行、第 j 列的元素为

$$[N(s)]_{ij}=\beta_{ij_0}s^n+\beta_{ij_1}s^{n-1}+\cdots+\beta_{ij_n}$$

$d(s)$为 $G(s)$各个元素分母的最小公倍式，具有如下的形式：

$$d(s)=\alpha_0s^n+\alpha_1s^{n-1}+\alpha_2s^{n-2}+\cdots+\alpha_n$$

输入参数定义为：

$$num:=\begin{bmatrix}N_{11}\\ \vdots\\ N_{r1}\\ \vdots\\ N_{1c}\\ \vdots\\ N_{rc}\end{bmatrix}$$

$$den:=[\alpha_0\alpha_1\alpha_2\cdots\alpha_n]$$

num 为 $N(s)$各个多项式元素的系数；

den 为 $d(s)$的各项系数；

r，c 为 $N(s)$的维数；

tf 为由 num 和 den 构成的传递函数矩阵。

输出参数定义为：

a,b,c,d：状态空间块控制器的系数矩阵；

ss：状态空间块控制器对应的系统变量。

4.3 鲁棒控制工具箱功能函数

在 MATLAB 鲁棒控制工具箱中提供了丰富的功能函数，用来处理各种复杂的数据，这些函数见表 4-6。

表 4-6 鲁棒控制工具箱功能函数列表说明

函数名	函数功能说明
aresolv	求解连续 Riccati 方程
daresolv	求解离散 Riccati 方程
riccond	求解连续 Riccati 条件数
driccond	求解离散 Riccati 条件数
blkrsch	计算实矩阵的 Schur 形式
cschur	计算矩阵的复的 Schur 形式

4.3.1 Riccati 方程求解

在鲁棒控制系统分析和设计中，许多问题往往可以归结为代数 Riccati 方程的求解。鲁棒控制工具箱的函数 aresolv 用于求解如下的连续代数 Riccati 方程：

$$A^T P + PA - PRP + Q = 0$$

1．aresolv

功能：求解连续代数 Riccati 方程

语法：[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r)

[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r,Type)

说明：输入参数定义为：

a,q,r：代数 Riccati 方程的矩阵 A，Q，R；

Type：设定求解方法；

Type='eigen'：特征向量方法；

Type='Schur'：Schur 向量方法；

输出参数定义为：

p：代数 Riccati 方程的解 p；

p1,p2：矩阵 P_1 和 P_2，满足 $P=P_2P_1^{-1}$；

lamp：闭环特征值；

perr：计算残差；

wellposed：当代数 Riccati 方程对应的 Hamilton 矩阵具有虚轴上的特征值时，取值为 FALSE；否则为 TRUE。

2．daresolv

功能：求解离散代数 Riccati 方程。

语法：[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r)

[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r,Type)

说明：输入参数定义为：

a,q,r：代数 Riccati 方程的矩阵 A，Q，R；

Type：设定求解方法；

Type='eigen'：特征向量方法；

Type='Schur'：Schur 向量方法；

输出参数定义为：

p：代数 Riccati 方程的解 p；

p1,p2：矩阵 P_1 和 P_2，满足 $P=P_2P_1^{-1}$；

lamp：闭环特征值；

perr：计算残差；

wellposed：当代数 Riccati 方程对应的 Hamilton 矩阵具有虚轴上的特征值时，取值为 FALSE；否则为 TRUE。

4.3.2 Riccati 方程条件数

1．riccond

功能：求解连续代数 Riccati 方程的条件数。

语法：[tot] = riccond(a,b,qrn,p1,p2)

说明：输入参数定义为：

a,b,qrn：代数 Riccati 方程的系数矩阵，对应的 Riccati 方程具有下面的形式：

$$A'P + PA - (PB + N)R^{-1}(B'P + N') + Q = 0$$

p1.p2：[p1；p2]张成 Hamilton 矩阵的稳定特征空间。

输出参数定义为：

tot：与连续代数 Riccati 方程有关的多个条件数构成的向量。

2．driccond

功能：求解离散代数 Riccati 方程的条件数。

语法：[tot] = driccond(a,b,q,r,p1,p2)

说明：a,b,q,r：离散代数 Riccati 方程的系数矩阵，对应的离散代数 Riccati 方程具有下面的形式：

$$A^T PA - P + Q - A^T PB(R + B^T PB)^{-1} B^T PA = 0$$

p1.p2：[p1：p2]张成 Hamilton 矩阵的稳定特征空间。

输出参数定义为：

tot：与离散代数 Riccati 方程有关的多个条件数构成的向量。

4.3.3 矩阵的 Schur 形式

1．blkrsch

功能：实矩阵的块有序 Schur 形式。

语法：[v,t,m] = blkrsch(a,Type,cut)

说明：输入参数定义为：

a：实的方阵；

Type：指定 Schur 形式的对角块的特征值排序：

- Type=1： $Re(\lambda_i(B_1)) < 0,\ Re(\lambda_i(B_2)) > 0$
- Type=2： $Re(\lambda_i(B_1)) > 0,\ Re(\lambda_i(B_2)) < 0$
- Type=3： $Re(\lambda_i(B_1)) > 0,\ Re(\lambda_i(B_2))$
- Type=4： $Re(\lambda_i(B_1)) < 0,\ Re(\lambda_i(B_2))$
- Type=5： $|\lambda_i(B_1)| > |\lambda_i(B_2)|$
- Type=6： $|\lambda_i(B_1)| < |\lambda_i(B_2)|$

cut：方阵 B_1 的维数。

输出参数的定义如下：

v：矩阵 V；

t：矩阵 A 的 Schur 形式；

m：A 的稳定特征值的数目。

2．cschur

功能：矩阵的有序复 Schur 形式。

语法：[v,t,m,swap] = cschur(a,Type)

说明：

输入参数定义为：

a：矩阵 A；

Type：指定复 Schur 形式的对角块的特征值排序：

- Type=1：　$Re(\lambda_i(T_1))<0,\ Re(\lambda_i(T_2))>0$
- Type=2：　$Re(\lambda_i(T_1))>0,\ Re(\lambda_i(T_2))<0$
- Type=3：特征值实部按降序排列；
- Type=4：特征值实部按升序排列；
- Type=5：特征值的模按降序排列；
- Type=6：特征值的模按升序排列。

输出参数的定义如下：

v：矩阵 V；

t：矩阵 A 的 Schur 形式；

m：A 的稳定特征值的数目。

4.4　多变量波特图

在 MATLAB 鲁棒控制工具箱中提供了计算和绘制多变量波特图的函数，见表 4-7。

表 4-7　多变量波特图函数列表说明

函 数 名	函 数 功 能 说 明
cgloci	连续系统特征增益/相位波特图
dcgloci	离散系统特征增益/相位波特图
dsigma	离散系统奇异值波特图
muopt	利用乘子尺度话方法计算结构奇异值上界
osborne	基于 Osborne 方法计算结构奇异值上界
perron, psv	基于 Osborne 方法计算结构奇异值上界
sigma	连续系统奇异值波特图
ssv	结构奇异值波特图

4.4.1　频率响应的特征增益/相位波特图

1．cgloci

功能：计算和绘制连续多变量系统特征增益波特图。

语法：[cg,ph,w] = cgloci(a,b,c,d)

[cg,ph,w] = cgloci(a,b,c,d,'inv')

[cg,ph,w] = cgloci(a,b,c,d,w)

```
[cg,ph,w] = cgloci(a,b,c,d,w,'inv')
[cg,ph,w] = cgloci(ss,…)
```

说明：输入参数的定义如下：

a,b,c,d：系统状态空间矩阵；

ss：系统状态空间模型变量；

'inv'：指定计算和绘制逆系统 $G(s)^{-1}$ 的特征增益相位波特图；

w：频率向量，用于指定计算特征增益和相位的频率点。

输出参数的定义如下：

cg：系统频率响应的特征增益向量；

ph：系统频率响应的特征相位向量；

w：频率向量。

当不指定输出变量时，函数将绘制系统的特征增益和相位的波特图。

【例 1】 考虑 2*2 的传递函数矩阵

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

G(s)的对角化分解如下：

$$G(s)=X\Lambda(s)X^{-1}=\begin{bmatrix}7 & -8\\ -6 & 7\end{bmatrix}\begin{bmatrix}\dfrac{1}{s+1} & 0\\ 0 & \dfrac{2}{s+2}\end{bmatrix}\begin{bmatrix}7 & 8\\ 6 & 7\end{bmatrix}$$

采用如下的程序绘制该系统的特征增益波特图，如图 4-8 所示。

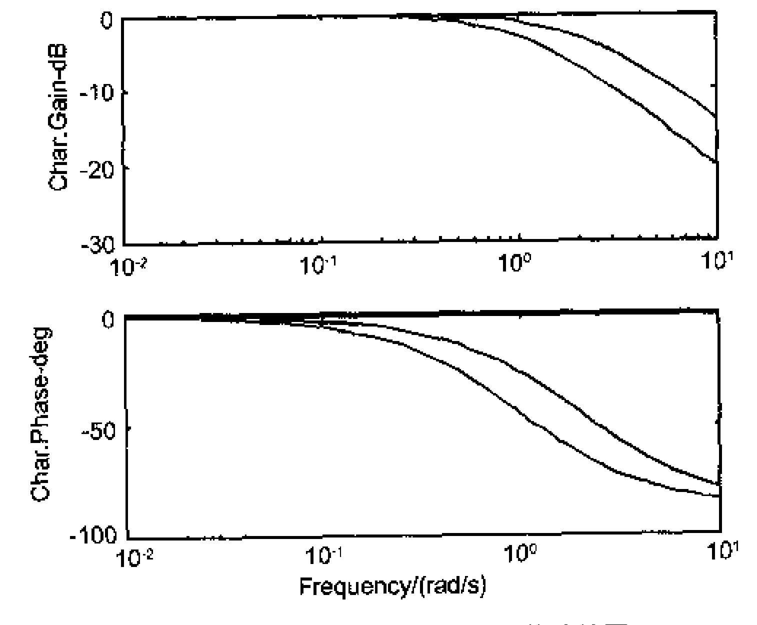


图 4-8 连续多变量系统特征增益波特图

```
num=[-47 2;56 0;-42 0;50 2];
den=[1 3 2];
[a,b,c,d]=tfm2ss(num,den,2,2);
cgloci(a,b,c,d)
```

2. dcgloci

功能：离散系统特征增益/相位波特图。

语法：[cg,ph,w] =dcgloci(a,b,c,d,Ts)

[cg,ph,w] =dcgloci(a,b,c,d,Ts,'inv')

[cg,ph,w] =dcgloci(a,b,c,d,Ts,w)

[cg,ph,w] =dcgloci(a,b,c,d,Ts,w,'inv')

[cg,ph,w] =dcgloci(ss,)

说明：输入参数的定义如下：

a,b,c,d：系统状态空间矩阵；

ss：系统状态空间模型变量；

'inv'：指定计算和绘制逆系统 $G(s)^{-1}$ 的特征增益相位波特图；

w：频率向量，用于指定计算特征增益和相位的频率点；

Ts：离散系统的采样周期。

输出参数的定义如下：

cg：系统频率响应的特征增益向量；

ph：系统频率响应的特征相位向量；

w：频率向量。

当不指定输出变量时，函数将绘制系统的特征增益和相位的波特图。

4.4.2 连续和离散系统的奇异值波特图

1. sigma

功能：连续系统奇异值波特图

语法：[sv,w] =sigma(a,b,c,d)

[sv,w] =sigma(a,b,c,d,'inv')

[sv,w] =sigma(a,b,c,d,w)

[sv,w] =sigma(a,b,c,d,w,'inv')

[sv,w] =sigma(ss,..)

说明：输入参数的定义如下：

a,b,c,d：系统状态空间矩阵；

ss：系统状态空间模型变量；

'inv'：指定计算和绘制逆系统 $G(s)^{-1}$ 的特征增益相位波特图；

w：频率向量，用于指定计算特征增益和相位的频率点。

输出参数的定义如下：

sv：系统的奇异值矩阵，该矩阵的每一列对应一个奇异值在各个频率点的幅值；

w：频率向量。

【例 1】 考虑 2*2 的传递函数矩阵

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

采用如下的程序绘制该系统的奇异值波特图，如图 4-9 所示。

```
num=[-47 2;56 0;-42 0;50 2];
den=[1 3 2];
[a,b,c,d]=tfm2ss(num,den,2,2);
w=logspace(-4,4);
sigma(a,b,c,d,w)
grid on
```

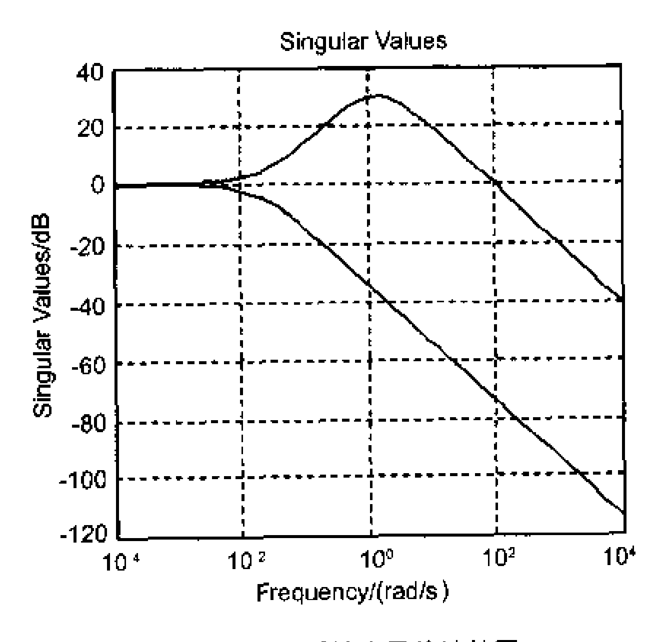


图 4-9 连续系统奇异值波特图

2. dsigma

功能：离散系统奇异值波特图。

语法：[sv,w] =dsigma(a,b,c,d,Ts)

[sv,w] =dsigma(a,b,c,d,Ts,'inv')

[sv,w] =dsigma(a,b,c,d,Ts,w)

[sv,w] =dsigma(a,b,c,d,Ts,w,'inv')

[sv,w] =dsigma(ss,..)

说明：在输入输出参数中，Ts 为采样周期；其他参数的定义与 sigma 函数相同。

【例 2】 计算和绘制多变量系统 G(s)在采样周期为 0.05s 时的离散奇异值波特图，如图 4-10 所示。其中

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

```
num=[-47 2;56 0;-42 0;50 2];
den=[1 3 2];
[a,b,c,d]=tfm2ss(num,den,2,2);
w=logspace(-4,4);
dsigma(a,b,c,d,0.05,w)
grid on
```

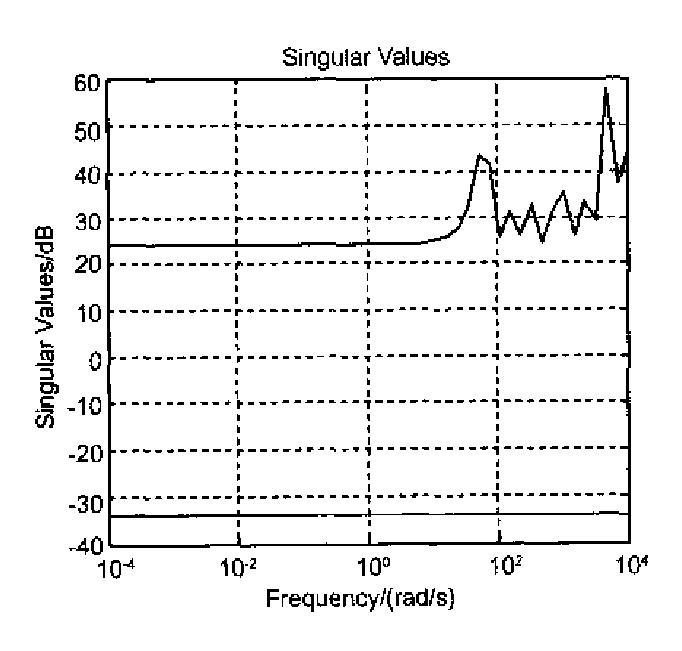


图 4-10　离散系统奇异值波特图

4.4.3　结构奇异值波特图

乘子尺度化方法示基于推广的 Popov 乘子理论。该理论由 Safonov 和 Lee 提出，并利用 Fan 和 Nekooie 的算法。乘子尺度化方法计算得到的具有 n 个不确定块的矩阵 A 的结构奇异值上界为下面的优化问题的解：

$$\min_{M\in \mathrm{M}} \quad \mu$$

约束条件

$$\tilde{A}_{scaled}+\tilde{A}^{*}_{scaled}\geqslant 0$$

其中，矩阵 A_s 为 A 的 Popov 乘子尺度化矩阵，即

$$A_{scaled}=\mu(I-\tilde{A}_{scaled})(I+\tilde{A}_{scaled})^{-1}$$

M 为不确定结构的各个块对角部分对应的广义 Popov 乘子构成的集合。

1. muopt

功能：利用乘子尺度话方法计算结构奇异值上界。

语法：[mu,ascaled,logm,x] = muopt(a)

[mu,ascaled,logm,x] = muopt(a,k)

说明：输入参数 a 为标称矩阵，k 为不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k = ones(p, 2)，即各个块均为 1*1 的矩阵。

输出参数的定义如下：

mu：矩阵 a 的结构奇异值上界；

ascaled：经过乘子尺度化后的矩阵 A，设乘子尺度化矩阵为 A_s，则经过尺度化后的矩阵 A 定义为

$$A_{scaled} = \mu(I - \tilde{A}_{scaled})(I + \tilde{A}_{scaled})^{-1}$$

logm：log (diag (M))；

x：矩阵的最小特征值的归一化特征向量。

2. osborne

功能：基于 Osborne 方法计算结构奇异值上界。

语法：[mu,ascaled,logd] = osborne(a)

[mu,ascaled,logd] = osborne(a,k)

说明：输入参数的定义为：

a：标称矩阵；

k：不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k = ones(p, 2)，即各个块均为 1*1 的矩阵。

输出参数的定义为：

mu：对角尺度化矩阵的最大奇异值，即矩阵 a 的结构奇异值上界的估计；

ascaled：尺度化矩阵 DAD^{-1}；

logd：对角尺度化矩阵 D 的对数矩阵。

【例 1】

```
A = eye(10);
A(1,10) = 100000;
[mu,Ascaled,logd] = osborne(A);
mu
```

MATLAB 返回：

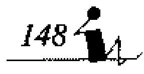
```
mu =
    1.0000
```

3. perrn

功能：基于 Osborne 方法计算结构奇异值上界。

语法：[mu] = perron(a)

[mu] = perron(a,k)



说明：输入参数定义为：

a：标称矩阵。

k：不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k = ones(p, 2)，即各个块均为 1*1 的矩阵。

输出参数的定义为：

mu：非负矩阵 F 的 Perron 特征值，其中 F 的元素为 A 的各个块矩阵的最大奇异值。该特征值作为 A 的结构奇异值上界的估计。

4．psv

功能：基于 Osborne 方法计算结构奇异值上界。

语法：[mu,ascaled,logd] = psv(a)

[mu,ascaled,logd] = psv(a,k)

说明：输入参数定义为：

a：标称矩阵；

k：不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k = ones(p, 2)，即各个块均为 1*1 的矩阵。

输出参数的定义为：

mu：A 的 Perron 最优尺度化矩阵的最大奇异值，即

$$mu = \bar{\sigma}[A_{scalcd}] = \bar{\sigma}[D_P A D_P^{-1}]$$

$$D_P = diag(d_1, d_2, \cdots, d_n)$$

$$d_i = \left|\sqrt{\frac{y_{P_i}}{x_{P_i}}}\right|$$

x_{Pi} 和 y_{Pi} 分别为 F 的 Perron 特征向量 x_P、y_P 的各个分量；

ascaled：矩阵 A 的 Perron 最优尺度化矩阵；

ogd：D_P 的对数矩阵。

【例 2】

```
A = eye(10);
A(1,10) = 100000;
[mu,Ascaled,logd] = psv(A);
s1 = max(svd(A));
[s1, mu]
```

MATLAB 返回：

```
ans =
   1.0e+005 *
      1.0000      0.0000
```

5．ssv

功能：结构奇异值波特图。

语法：[mu,logd] = ssv(a,b,c,d,w)

```
[mu,logd] = ssv(a,b,c,d,w,k)
[mu,logd] = ssv(a,b,c,d,w,k,opt)
[mu,logd] = ssv(ss,)
```

说明：函数 ssv 能够调用函数 osborne、perron、psv 和 muopt 四个函数之一来计算如下的传递函数矩阵的结构奇异值上界并绘制相应的波特图。

$$G(j\omega) = C(j\omega I - A)^{-1}B + DB$$

其中矩阵 A、B、C、D 为系统状态空间矩阵。

输入参数的定义如下：

a,b,c,d：标称系统的状态空间矩阵；

w：频率向量；

k：不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k = ones(p, 2)，即各个块均为 1*1 的矩阵。

opt：选择计算结构奇异值上界的方法，opt 可为下列取值之一：

- 'osborne'：对角尺度化方法；
- 'psv'：perron 最优对角尺度化方法；
- 'perron'：perron 特征值方法；
- 'muopt'：Popov 乘子尺度化方法。

ss：系统状态空间模型变量。

输出参数定义如下：

mu：系统在各个频率点的结构奇异值上界值；

logd：最优对角尺度化矩阵 $D(j\omega)$的对数波特图，如果不确定块均为复的，则 $D(j\omega)$为实的；如果不确定块同时具有实参数摄动和复摄动，则 logd 通常为复的，并包括最优乘子尺度的平方根波特图。

4.5 矩阵因子化技巧

MATLAB 鲁棒控制工具箱提供的关于矩阵因子化的函数见表 4-8。

表 4-8 矩阵因子化技巧函数列表说明

函数名	函数功能说明
iofr(iofc)	内外因子化
Sfl(sfr)	计算左(右)谱因子

1．iofr(iofc)

对于稳定的传递函数矩阵 G，设行维数为 m，列维数为 n，且 m≥n，其内外因子化形式为：

$$G = [\theta \quad \theta^{\perp}]\begin{bmatrix} M \\ 0 \end{bmatrix}$$

函数 iofr 用于计算矩阵 G(m≥n)内外因子化。

功能：矩阵的内外因子化。

语法：[ain,,ainp,,aout,] = iofr(a,b,c,d)

　　　[ssin,ssinp,ssout] = iofr(ss)

说明：输入参数定义为：

a,b,c,d：传递函数矩阵的状态空间实现；

ss：ss=mksys(a,b,c,d, 'ss')。

输出参数定义为：

ain,bin,cin,din：θ 的状态空间实现；

ainp,binp,cinp,dinp：$\theta^{\perp}$的状态空间实现；

aout,bout,cout,doutM：M 的状态空间实现；

ssin=mksys(ain,bin,cin,din)；

ssinp=mksys(ainp,binp,cinp,dinp)；

ssout=mksys(aout,bout,cout,dout)。

当传递函数矩阵 G 的行维数 m 小于列维数 n 时，函数 iofc 用于计算 G 的内外因子化，采用的算法是首先调用函数 iofr 计算 G 的转置矩阵的内外因子化，然后再变换得到 G 的内外因子化矩阵，其使用格式与 iofr 相同。

2. sfl(sfr)

功能：计算左（右）谱因子。

语法：[am,bm,cm,dm] = sfl(a,b,c,d)

　　　[am,bm,cm,dm] = sfr(a,b,c,d)

　　　[ssm] = sfl(ss)

　　　[ssm] = sfr(ss)

说明：输入参数定义为：

a,b,c,d：传递函数矩阵的状态空间实现；

ss：ss=mksys(a,b,c,d, 'ss')。

输出参数定义为：

am,bm,cm,dm：$M(s)$的状态空间实现；

ssm=mksys(am,bm,cm,dm, 'ss')。

4.6　模型降阶方法

MALTAB 鲁棒控制工具箱提供了对模型进行降阶处理的函数，见表 4-9。

表 4-9　模型降阶方法函数列表说明

函 数 名	函 数 功 能 说 明
balmr	均衡模型降阶
bstschml, bstschmr	Schur 相对误差降阶

(续)

函 数 名	函 数 功 能 说 明
imp2ss	脉冲响应向状态空间模型的转换
obalreal	有序均衡实现
ohklmr,ohkapp	最优 Hankel 最小阶逼近降阶
schmr	Schur 模型降阶

4.6.1 Schur 相对误差模型降阶方法

给定一个 n 阶稳定对象

$$G(s):=\left[\begin{array}{c|c}A & B\\ \hline C & D\end{array}\right]$$

Schur 相对误差模型降阶方法计算得到的 k 阶降阶模型 $\bar{G}(s)$ 同时满足下面两个误差上界条件，即

乘性误差上界

$$\left\|\bar{G}^{-1}(G-\bar{G})\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

相对误差上界

$$\left\|\bar{G}^{-1}(G-\bar{G})\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

其中，σ 为全通相位矩阵 $(W*(s))^{-1}G(s)$ 的 Hankel 奇异值，$W(s)$ 为矩阵 $\Phi = G(s)G^{\mathrm{T}}(-s)$ 的左谱因子，即 $\Phi = W^{\mathrm{T}}(-s)W(s)$。

函数 bstschml 和 bstschmr 分别用于计算 Schur 相对误差模型降阶以及其对偶问题，所谓 Schur 相对误差模型降阶的对偶问题时指具有如下的误差上界条件的模型降阶问题

乘性误差上界

$$\left\|\bar{G}^{-1}(G-\bar{G})\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

相对误差上界

$$\left\|\bar{G}^{-1}(G-\bar{G})\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$$

1. bstschmr

功能：Schur 相对误差模型降阶。

语法：[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type)

[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no)

[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no,info)
[ssred,aug,svh] = bstschmr(SS,Type)
[ssred,aug,svh] = bstschmr(SS,Type,no)
[ssred,aug,svh] = bstschmr(SS,Type,no,info)

说明：输入参数定义为：

A,B,C,D：原系统的状态空间模型；

Type：指定参数类型，no 为相应参数，Type 可以取值为：

- Type = 1，　no = k.，k 为降阶模型的阶数；
- Type = 2，no = tol，tol 为相对误差的分贝值，使得对于使用频率，下式成立：

$$\bar{G}(j\omega) \subset G(j\omega) \pm tol$$

- Type = 3，显示 Hankel 奇异值向量 svh 和模型阶数 k；

info：指定误差指标的类型，缺省值为 right，即完成 Schur 相对误差模型降阶，当 info=left 时，调用函数 bstschml，完成对偶问题的求解：

ss：ss=mksys(A,B,C,D, 'ss')。

输出参数定义为：

ared,bred,cred,dred：降阶模型的状态空间矩阵；

aug：被删除的状态以及相对误差界，其中

- aug(1,1)：被删除的状态；
- aug(2,2)：相对误差界；

svh：Hankel 奇异值向量。

2．bstschml

功能：Schur 相对误差降阶。

语法：[ared,bred,cred,dred,aug,svh] = bstschml(A,B,C,D,Type)
[ared,bred,cred,dred,aug,svh] = bstschml(A,B,C,D,Type,no)
[ared,bred,cred,dred,aug,svh] = bstschml(A,B,C,D,Type,no,info)
[ssred,aug,svh] = bstschml(SS,Type)
[ssred,aug,svh] = bstschml(SS,Type,no)
[ssred,aug,svh] = bstschml(SS,Type,no,info)

说明：参数定义与函数 bstschmr 基本相同，只是输入参数 info 取值只能为 left。当函数 bstschmr 的输入参数 info 为 left 时，其功能与函数 bstschml 完全相同。

4.6.2　均衡模型降阶

1．balmr

功能：基于截断的均衡模型降阶。

语法：[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Type)
[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Type,aug)
[ssm,totbnd,svh] = balmr(ss,Type,aug)

说明：输入参数定义为：

a,b,c,d：系统的状态空间矩阵；

Type：指定参数类型，aug 为相应参数，Type 可以取值为：

- Type = 1，aug= k.，k 为降阶模型的阶数；
- Type = 2，aug= tol，计算一个 k 阶模型，使误差无穷范数上界小于 tol；
- Type = 3，显示 Hankel 奇异值向量和模型阶数 k。

输出参数定义为：

am,bm.cm.dm：降阶模型的均衡状态空间实现；

totbnd：模型误差的无穷范数上界；

svh：$G(j\omega)$的稳定和不稳定的 Hankel 奇异值。

2. obalreal

功能：系统状态空间模型的有序均衡实现。

语法：[abal,bbal,cbal,g,t] = obalreal(a,b,c)

说明：输入参数定义为：

a,b,c：原系统的状态空间矩阵。

输出参数定义为：

abal,bbal,cbal：系统的有序均衡实现；

g：均衡系统的 Gram 矩阵；

t：均衡实现的状态变换矩阵。

该函数要求原系统是最小实现，否则算法将无法成功执行。

3. schmr

功能：Schur 均衡模型降阶。

语法：[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type)

[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type,aug)

[ssm,totbnd,svh] = schmr(ss,Type,aug)

说明：输入参数定义为：

a,b,c,d：系统的状态空间矩阵；

Type：指定参数类型，aug 为相应参数，Type 可以取值为：

- Type = 1，aug= k.，k 为降阶模型的阶数；
- Type = 2，aug= tol，计算一个 k 阶模型，使误差无穷范数上界小于 tol；
- Type = 3，显示 Hankel 奇异值向量和模型阶数 k。

输出参数定义为：

am,bm.cm.dm：降阶模型的均衡状态空间实现；

totbnd：模型误差的无穷范数上界；

svh：$G(j\omega)$的稳定和不稳定的 Hankel 奇异值。

4.6.3 最优 Hankel 最小逼近降阶

1. ohkapp

功能：计算稳定对象的最优 Hankel 最小阶逼近降阶模型。

语法：[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type)



[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type,in)

[ssx,ssy,] = ohkapp(ss,)

说明：输入参数定义为：

a,b,c,d：系统的状态空间矩阵；

Type：指定参数类型，Type 可以取值为：

- Type = 1，in= k.，k 为降阶模型的阶数；
- Type = 2，in= tol，计算一个 k 阶模型，使误差无穷范数上界小于 tol；
- Type = 3，显示 Hankel 奇异值向量和模型阶数 k；

输出参数定义为：

ax,bx.cx.dx：降阶模型的均衡状态空间实现；

ay,by,cy,dy：反因果系统 Gy(s)的状态空间实现，并满足

$$\|G-G_x-G_y\|_\infty \leqslant \sigma_{k+1}$$

参数 aug 的各个分量定义为

- aug(1,1) = σ_1；
- aug(1,2) = 删除的状态数；
- aug(1,3) = totbnd；
- aug(4:4+n-1) = [$\sigma_1,\sigma_2,\ldots,\sigma_n$]。

2. ohklmr

功能：计算不稳定对象的最优 Hankel 最小阶逼近降阶模型。

语法：[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type)

[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type,in)

[ssm,] = ohklmr(ss,..)

说明：输入参数定义为：

a,b,c,d：系统的状态空间矩阵；

Type：指定参数类型，Type 可以取值为：

- Type = 1，aug= k.，k 为降阶模型的阶数；
- Type = 2，aug= tol，计算一个 k 阶模型，使误差无穷范数上界小于 tol；
- Type = 3，显示 Hankel 奇异值向量和模型阶数 k；

输出参数定义为：

am,bm.cm.dm：降阶模型的状态空间实现；

totbnd：模型误差的无穷范数上界；

svh：$G(j\omega)$的稳定和不稳定投影的 Hankel 奇异值。

4.7　鲁棒控制箱综合方法

表 4-10 中列出了 MATLAB 鲁棒控制工具箱中对一些综合问题的处理函数。

表 4-10 鲁棒控制工具箱综合方法函数列表说明

函数名	函数功能说明
h2lqg, dh2lqg	连续（离散）系统 H^2 最优控制器综合
hinf, linf, dhinf	连续（离散）系统 H_∞ 最优控制器综合
Hinfopt	基于 γ 迭代方法计算 H_∞ 最优控制器
normh2, normhinf	计算系统的 H^2 和 H_∞ 范数
Lqg	计算 LQG 优化控制器
ltru, ltry	LQG 环路传输恢复
musyn, fitd, augd	γ 综合
youla	Youla 参数化

4.7.1 离散和连续情形的 H^2 综合

考虑如图 4-11 的闭环系统，H^2 优化控制问题即：为一个增广系统 $P(s)$

$$P(s):=\left[\begin{array}{cc|c} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

寻找一个稳定的正反馈控制器，使得闭环传递函数矩阵的 H^2 范数满足：

$$\min_{F(S)}\left\|T_{y_1u_1}\right\|_2:=\min_{F(S)}\left(\frac{1}{\pi}\right)\int_0^\infty\left(\sigma ace(T^*_{y_1u_1}(j\omega)T_{y_1u_1}(j\omega))d\omega\right)$$

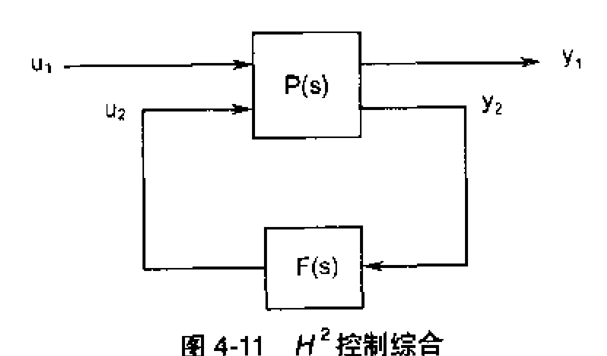


图 4-11 H^2 控制综合

求解上述问题的方法是将其转化为相应的 LQG（*Linear-Quadratic Gaussian*）优化控制问题，等价的优化指标为

$$J_{LOG}=\lim_{T\to\infty}E\left\{\frac{1}{T}\int_0^T y_1^T y_1\,dt\right\}=$$

$$\lim_{T\to\infty}E\left\{\frac{1}{T}\int_0^T[x^T\quad u_2^T]\begin{bmatrix} Q & N_C \\ N_C^T & R\end{bmatrix}\begin{bmatrix} x \\ u_2\end{bmatrix}dt\right\}=$$

$$\lim_{T\to\infty}E\left\{\frac{1}{T}\int_0^T[x^T\quad u_2^T]\begin{bmatrix} C_1^T \\ D_{12}^T\end{bmatrix}[C_1\quad D_{12}]\begin{bmatrix} x \\ u_2\end{bmatrix}dt\right\}$$

对应的白噪声协方差矩阵为

$$E\left\{\begin{bmatrix}\xi(t)\\ \theta(t)\end{bmatrix}[\xi(\tau)\quad \theta(\tau)^T]\right\}=\begin{bmatrix}\Xi & N_f\\ N_f^T & \Theta\end{bmatrix}\begin{bmatrix}B_1\\ D_{21}\end{bmatrix}\begin{bmatrix}B_1^T & D_{21}^T\end{bmatrix}\delta(t-\tau)$$

$$=\begin{bmatrix}B_1B_1^T & B_1D_{21}^T\\ D_{21}B_1^T & D_{21}D_{21}^T\end{bmatrix}\delta(t-\tau)$$

转化为 LQG 优化问题后，根据分离原理，得到：

- Kalman 滤波器

$$\dot{\hat{x}}=A\hat{x}+B_2u_2+K_f(y_2-C_2\hat{x}-D_{22}u_2)$$

$$K_f=(\Sigma C_2^T+N_f)\Theta^{-1}=(\Sigma C_2^T+B_1D_{21}^T)(D_{21}D_{21}^T)^{-1}$$

其中 $\Sigma=\Sigma^T$，且满足：

$$\Sigma A^T+A\Sigma-(\Sigma C_2^T+N_f)\Theta^{-1}(C_2\Sigma+N_f^T)+\Xi=0$$

- 全状态反馈

$$u_2=K_C\hat{X}$$

$$K_C=R^{-1}(B_2^TP+N_C^T)=(D_{12}^TD_{12})^{-1}(B_2^TP+D_{12}^TC_1)$$

其中 $P=P^T$ 且满足：

$$A^TP+PA-(PB_2+N_C)R^{-1}(B_2^TP+N_C^T)+Q=0$$

1．h2lqg

功能：连续系统 H_2 最优控制器综合。

语法：[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = h2lqg(A,B1,B2,,D22)

[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = h2lqg(A,B1,B2,,D22,aretype)

[sscp,sscl] = h2lqg(TSS)

[sscp,sscl] = h2lqg(TSS,aretype)

说明：输入参数定义为：

A、B1、B2、C1、C2、D11、D12、D21、D22：双端口连续系统的状态空间矩阵；

TSS：双端口系统定义的树型模型变量；

aretype：用于指定求解代数 Ricccati 方程的方法，其取值如下：

- aretype='eigen'：特征向量方法，缺省值；
- aretype='Schur'：Schur 向量方法。

输出参数定义为：

acp,bcp,ccp,dcp：状态反馈传递函数矩阵的状态空间实现；

acl,bcl,ccl,dcl：闭环系统的状态空间矩阵；

sscpl：状态反馈矩阵对应的状态空间模型变量；

sscel：闭环系统的状态空间模型变量。

2．dh2lqg

功能：离散系统 H_2 最优控制器综合。

语法：[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] =dh2lqg(A,B1,B2,,D22)

[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] =dh2lqg(A,B1,B2,,D22,aretype)

[sscp,sscl] =dh2lqg(TSS)

[sscp,sscl] =dh2lqg(TSS,aretype)

说明：输入参数与输出参数的定义与 h2lqg 相同。

对于上述两个函数需要加以说明的有以下几点：

- (A，B2，C2)必须是可镇定的和可测的；
- 对连续系统的 H_2 优化问题，要求矩阵 D11 必须为 0，如果输入参数中的 D11 不为 0，函数 h2lqg 将自动地把 D11 作为零矩阵处理；
- D12 和 D21 的转置必须是列满秩的。

4.7.2 离散和连续情形的 H_∞ 综合

对于双端口系统 $P(s)$

$$P(s):=\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

找出一个稳定控制器 $F(s)$，使得闭环传递函数适合无限范数不等式：

$$\left\|T_{y_1u_1}\right\|_\infty \stackrel{\Delta}{=} \sup_\omega \sigma_{\max}(T_{y_1u_1}(j\omega)) < 1$$

对应的闭环系统如图 4-12 所示。

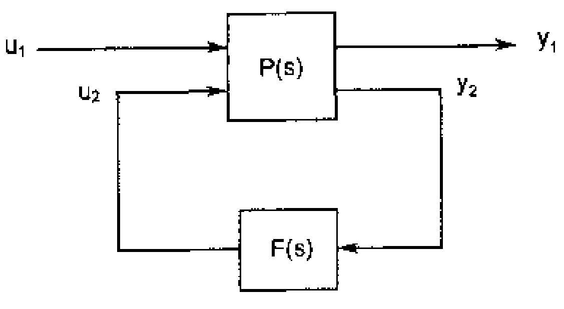


图 4-12　闭环控制系统

hinf 和 dhinf，linf

功能：H_∞最优控制综合(离散的和连续的)。

语法：linf

Inputs: A, B1, B2, C1, C2, D11, D12, D21, D22

Outputs: acp, bcp, ccp, dcp, acl, bcl, ccl, dcl

[acp,,acl,,hinfo,ak,,dk22] = (d)hinf(A,,D22)

[acp,,acl,,hinfo,ak,,dk22] = (d)hinf(A,,D22,au,,du)

[acp,,acl,,hinfo,ak,,dk22] = ... (d)hinf(A,,D22,au,,du,verbose)

[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP)

[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP,ssu)

[sscp,sscl,hinfo,tssk] = (d)hinf(TSSP,ssu,verbose)

说明：linf 的输入参数 A, B1, B2, C1, C2, D11, D12, D21, D22 为增广对象的状态空间矩阵。

输出参数定义为：acp, bcp, ccp, dcp 为 L_∞控制器，acl, bcl, ccl, dcl 为闭环控制系统状态空间矩阵。

hinf 和 dhinf 输入参数定义为：

A,…,D22：双端口系统的状态空间矩阵；

au,bu,cu,du：用于参数化 H_∞控制器的传递函数的状态空间实现；

verbose：用于指定计算过程中是否显示有关结果，如果为 1，则显示；为 0 则不显示；

输出参数定义为：

acp,bcp,ccp,dcp：一个控制器特解的状态空间实现；

ac1,bc1,cc1,dc1：闭环系统的状态空间实现；

ak,,dk22：全解 H_∞控制器的参数化实现；

hinfo：有关计算过程的信息。

4.7.3　H_∞综合的γ迭代方法

- hinfopt

功能：H_∞综合的γ迭代方法。

语法：[gamopt,acp,,dcp,acl,,dcl] = hinfopt(A,,D22)

[gamopt,acp,,dcp,acl,,dcl] = hinfopt(A,,D22,gamind)

[gamopt,acp,,dcp,acl,,dcl] = hinfopt(A,,D22,gamind,aux)

[gamopt,sscp,sscl] = hinfopt(tss)

[gamopt,sscp,sscl] = hinfopt(tss,gamind)

[gamopt,sscp,sscl] = hinfopt(tss,gamind,aux)

说明：输入参数：

A,,D22：增广对象的状态空间矩阵；

gamind：指定被γ尺度化的输出通道，缺省为：[1:n]

aux：用于指定γ迭代方法停止条件和γ迭代的γ值范围，格式如下：

aux=[tol maxgam mingam]

其中 tol 指定γ迭代方法停止条件，maxgam mingam 分别指定γ迭代最大和最小的γ值；

tss：增广对象的状态空间模型变量。

输出参数：

gamopt：迭代得到的最优γ值；

acp,,dcp：控制器的状态空间矩阵；

acl,,dcl：闭环系统的状态空间矩阵；

sscp=mksys(acp,bcp,ccp,dcp,'ss')；

sscl=mksys(acl,bcl,ccl,dcl, 'ss')。

【例 1】 以为三个使用 hinfopt 函数处理 SISO 系统的例子。

系统对象为：

$$G(s)=\frac{s-1}{s-2}$$

(1) 混合灵敏度指标，没有 W_3

$$W_1 = \frac{0.1(s+1000)}{100s+1},\ W_2 = 0.1$$

```
[ag,bg,cg,dg] = tf2ss([1 -1],[1 -2]);
ssg = mksys(ag,bg,cg,dg);
w1 = [0.1*[1 100];[100 1]]; w2 = [0.1;1]; w3 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,[1:2],[0.001,1,0]);
```

迭代显示经过 12 次迭代后得到 γ 最优值为 1.5146。

(2) 无 W_1

```
w1 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

此时得到 γ 最优为 2.5，$F(s) = -4/3$

(3) 无 W_2

$$W_1 = \frac{s+1}{10s+1}$$

```
w1 = [1 1;10 1]; w2 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

此时 $F(s) = \infty$，最优 γ 为 11/6。

4.7.4 H^2 和 H_∞ 范数

1．normh2

功能：计算系统的 H^2 范数。

语法：[h2n] = normh2(a,b,c,d)

[h2n] = normh2(ss)

说明：计算系统 G(s)=(A,B,C,D)的 H^2 范数，ss 为状态空间对象。

2．normhinf

功能：计算系统的 H_∞ 范数。

语法：[hinfn] = normhinf(a,b,c,d,aux)

[hinfn] = normhinf(a,b,c,d)

[hinfn] = normhinf(ss,aux)

[hinfn] = normhinf(ss)

说明：计算系统 G(s)=(A,B,C,D)的 H_∞ 范数，ss 为状态空间对象。可选参数 aux=[tol gammax gammin]，tol 为停止搜索的条件，缺省为 0.001，gammax 和 gammin 为估计的范数的最大最小值。缺省为

$$\text{gammin}=\max[\bar{\sigma}(D),\bar{\sigma}_H(G)]$$

$$\text{gammax}=\bar{\sigma}(D)+2\sum_{i=1}^{n}\bar{\sigma}_{H_i}(G)$$

其中 $\sigma_{Hi}(G)$为矩阵 $G(s)$ 的 Hankel 奇异值。

4.7.5 LQG 优化控制综合

考虑如下状态空间模型：

$$\dot{x} = Ax + Bu + \xi$$
$$y = Cx + Du + \theta$$

寻找如下的 LQG 优化控制指标：

$$J_{LQG} = \lim_{T\to\infty} E\left\{\int_0^T [x^T \quad u^T]\begin{bmatrix} Q & N_C \\ N_C^T & R \end{bmatrix}\begin{bmatrix} x \\ u \end{bmatrix} \mathrm{d}t\right\}$$

其中系统噪声 ξ 和测量噪声 θ 为高斯型的白噪声，并具有如下的协方差矩阵：

$$E\left\{\begin{bmatrix} \xi(t) \\ \theta(\tau) \end{bmatrix}[\xi(t)\theta(\tau)]^T\right\} = \begin{bmatrix} \Xi & N_f \\ N_f^T & \Theta \end{bmatrix}\delta(t-\tau)$$

对应的闭环控制系统如图 4-13 所示。

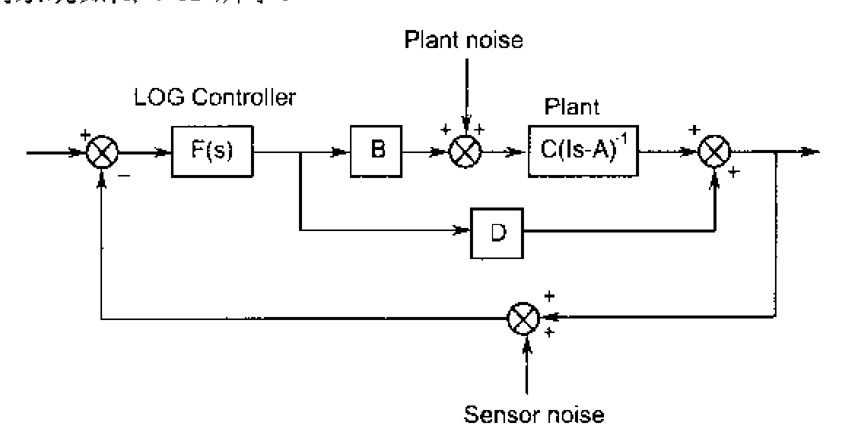


图 4-13　LQG 闭环控制系统

根据分离原理，LQG 优化控制问题就是求解 Kalman 滤波问题和全状态反馈问题的综合，最终求得的负反馈控制器具有如下形式：

$$u = -F(s)\, y$$

$$F(s) := \left[\begin{array}{c|c} A - K_f C_2 - B_2 K_c + K_f D_{22} K_C & K_f \\ \hline K_c & 0 \end{array}\right]$$

- lqg

功能：LQG 优化控制综合。

语法：[af,bf,cf,df] = lqg(A,B,C,D,W,V)

[ssf] = lqg(ss,w,v)

说明：输入参数：

A,B,C,D 为开环系统的状态空间矩阵；

W 和 V 具有如下形式：

$$W=\begin{bmatrix} Q & N_c \\ N_c^T & R \end{bmatrix};\ V=\begin{bmatrix} \Xi & N_f \\ N_f^T & \Theta \end{bmatrix}$$

输出参数：

af,bf,cf,df 为反馈控制器的状态空间矩阵；

ssf：反馈控制器对应的状态空间模型变量。

4.7.6 LQG 回路传输恢复

1. ltru

功能：基于控制量的全状态回路传输恢复。

语法：[af,bf,cf,df,svl] = ltru(A,B,C,D,Kc,Xi,Th,r,w)

[ssf,svl] = ltru(ss,Kc,Xi,Th,r,w,svk)

说明：输入参数：

A,B,C,D：开环系统状态空间矩阵；

Kc：LQR 全状态反馈增益矩阵；

Xi：实际对象的噪声协方差矩阵；

Th：观测噪声协方差矩阵；

r：恢复增益向量；

w：波特图的频率向量；

svk：全状态反馈闭环传递函数的频率响应。

输出参数：

af,bf,cf,df：闭环系统控制器的状态空间实现，传递函数为：

$$F(s)=K_c(Is-A+BK_c+K_fC-K_fDK_c)^{-1}K_f$$

其中，K_f 和 K_c 分别为滤波增益器和全状态反馈增益矩阵；

svl：闭环系统的奇异值。

2. ltry

功能：基于输出量的全状态回路传输恢复。

语法：[af,bf,cf,df,svl] = ltry(A,B,C,D,Kf,Q,R,q,w)

[ssf,svl] = ltry(ss,Kf,Q,R,q,w,svk)

说明：输入参数 A,B,C,D：开环系统状态空间矩阵；

Kf：滤波器增益矩阵；

Q：LQG 性能指标加权矩阵；

R：LQG 性能指标加权矩阵；

q：恢复增益向量；

w：波特图的频率向量；

svk：全状态反馈闭环传递函数的频率响应。

输出参数：

af,bf,cf,df：闭环系统控制器的状态空间实现，传递函数为：

$$F(s)=K_c(Is-A+BK_c+K_fC-K_fDK_c)^{-1}K_f$$

其中，K_f和 K_c 分别为滤波增益器和全状态反馈增益矩阵；

svl：闭环系统的奇异值。

4.7.7　μ 综合

1. musyn

功能：μ 综合。

语法：[acp,,dcp,mu,logd,ad,,dd,gam] = musyn(A,B1,B2,,D22,w)

[acp,,dcp,mu,logd,ad,,dd,gam] = musyn(A,B1,B2,,D22,w,gammaind,aux,logd0,n,blksz,flag)

[sscp,mu,logd,ssd,gam] = musyn(tss,w)

[sscp,mu,logd,ssd,gam] = musyn(tss,w,gammaind,aux,logd0,n,blksz,flag)

说明：给定一个二端口的状态空间模型：

$$P(s):=\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

函数 musyn 使用 D-F 迭代算法来计算控制器 $F(s)$

$$F(s):=\begin{bmatrix} A_{cp} & B_{cp} \\ C_{cp} & D_{cp} \end{bmatrix}$$

和对角化尺度矩阵 $D(s)$

$$D(s)=diag(d_1(s)I_{k_1},\text{L}\ ,d_n(s)I_{k_n})$$

使得它们满足鲁棒性能指标：

$$\left\|DT_{y_1u_1}D^{-1}\right\|_\infty<1$$

函数输入参数定义：

A,B1,B2,,D22：开环双端口对象；

w：计算结构奇异值得向量；

gammaind，aux：指定 H_∞优化参数（见 hinfopt）；

logd：指定 $D(s)$得初始值；

n,blksz,flag：指定函数 fitd 的参数；

输出参数：

acp,,dcp：控制器的状态空间实现；

mu：最优的 μ 值；

logd：$D(s)$的对数幅值频率特性；

ad,,dd：$D(s)$的状态空间实现；

gam：H_∞优化得到的最终 γ 值。

【例 1】 以下为解决一个简单 μ 综合问题的 MATLAB 程序：

```
a=2;
b1=[.1,-1];
```

```
b2=-1;
c1=[1;.01];
d11=[.1,.2;.01,.01];
d12=[1; 0];
c2=1;
d21=[0,1];
d22=3;
tss=mksys(a,b1,b2,c1,c2,d11,d12,d21,d22,'tss');
w = logspace(-2,1);%频率向量
%开始μ综合 D-F 迭代
[sscp,mu,logd0] = musyn(tss,w);
%显示最优μ值，见图 4-14
loglog(w,mu);
%采用频率相关的D(s)改善结果
[sscp,mu1,logd1] = musyn(tss,w,[ ],[ ],logd0,1);
%显示最优μ值，见图 4-15
loglog(w,mu,w,mu1);
```

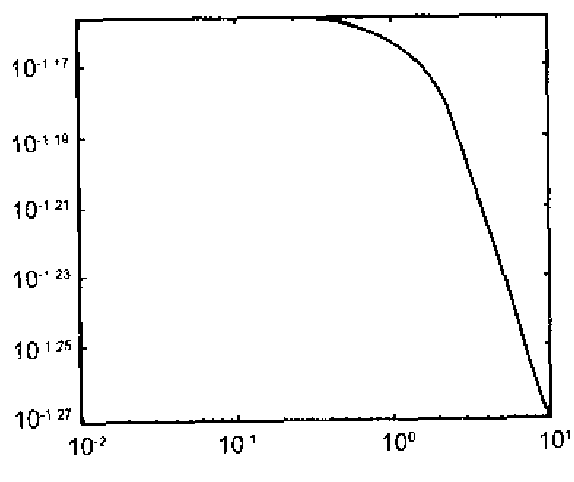


图 4-14　最优结构奇异值曲线

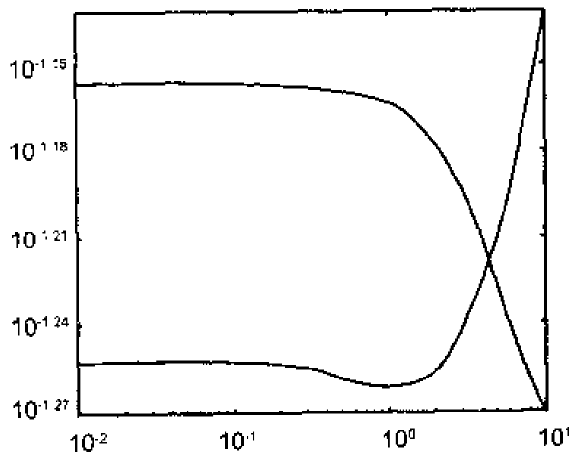


图 4-15　改善后最优结构奇异值与改善前比较

2. fitd

功能：多变量波特值的状态空间实现。

语法：
```
[ad,bd,cd,dd,logdfit] = fitd(logd,w)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n,blksz)
[ad,bd,cd,dd,logdfit] = fitd(logd,w,n,blksz,flag)
[ssd,logdfit] = fitd()
```

说明：fitd 返回对角传递函数矩阵的极小相位状态空间的实现，输入参数 logd 为矩阵，其行为在频率 w 处取值的对数波特点，可选参数：

n：包含状态空间对角元素逼近阶次的向量，缺省为 0；



blksz：对角块的大小，缺省为 1；

flag：缺省为 1，表示显示波特图；

fitd 使用信号处理工具箱的标准函数 yulewalk 实现多变量波特值；

3. augd

功能：增广两端口的系统。

语法：[AD,BD1,BD2,CD1,CD2,DD11,DD12,DD21,DD22] = augd(a,b1,b2,c1,c2,d11,d12,d21,d22,ad,bd,cd,dd)

[TSSD] = augd(TSS,ssd)

说明：augd 使用 H_∞综合通过对角缩放增广两端口的系统，如图 4-16 所示。

$$D(s)P(s)D(s)^{-1}=\begin{bmatrix} DP_{11}D^{-1} & DP_{12} \\ P_{21}D^{-1} & P_{22} \end{bmatrix}$$

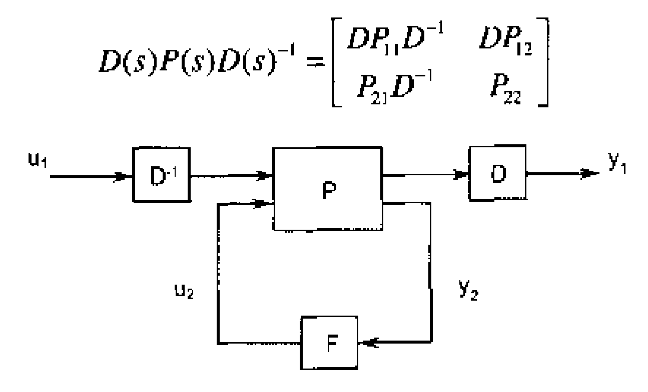


图 4-16　对角缩放增广系统

输入参数：

a,b1,b2,c1,c2,d11,d12,d21,d22 为增广前系统状态空间矩阵，对应的状态空间模型变量为 TSS；

ad,bd,cd,dd 用于参数化 H_∞综合控制器的传递函数的状态空间实现，对应模型变量为 ssd；

输出参数 AD,BD1,BD2,CD1,CD2,DD11,DD12,DD21,DD22 增广对象的状态空间矩阵，对应的输出状态空间模型变量为 TSSD。

4.7.8　youla 参数化

- youla

功能：计算双端口开环系统的所有可实现的稳定闭环系统 youla 参数化形式。

语法：youla

Inputs: A, B1, B2, C1, C2, D11, D12, D21, D22

Outputs: at11, bt11, ct11, dt11

at12, bt12, ct12, dt12, at1p, bt1p, ct1p, dt1p

at21, bt21, ct21, dt21, at2p, bt2p, ct2p, dt2p

kx, x, ky, y, f, h

说明：给定增广的状态空间实现：

$$\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

youla 计算计算 LQG 控制器 $K(s)$使得如图 4-17 所示的闭环系统 $T(s)$具有如下形式：

$$T(s):=\begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & T_{22}(s) \end{bmatrix}=\begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & 0 \end{bmatrix}$$

其中

$$T^{T}_{12}(-s)T_{12}(s)=I$$

$$T_{21}(s)T^{T}_{21}(-s)=I$$

则闭环传递函数为

$$T_{y_1u_1}=T_{11}(s)+T_{12}(s)Q(s)T_{21}(s)$$

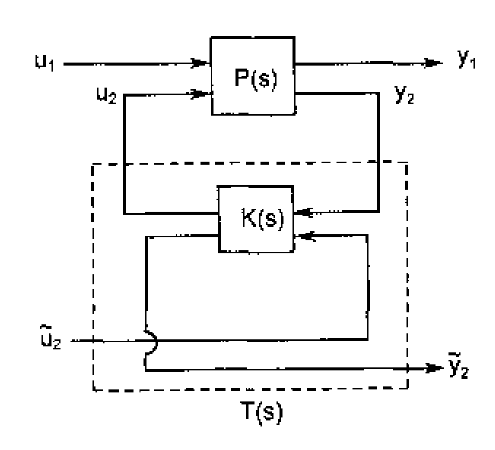


图 4-17　youla 参数化

输入参数定义：

A, B1, B2, C1, C2, D11, D12, D21, D22：双端口系统的状态空间矩阵。

输出参数定义：

at11, bt11, ct11, dt11：传递函数 T_{11} 的状态空间实现；

at12, bt12, ct12, dt12：传递函数 T_{12} 的状态空间实现；

at1p, bt1p, ct1p, dt1p：传递函数 $T_{12}^{\perp}$的状态空间实现；

at21, bt21, ct21, dt21：传递函数 T_{21} 的状态空间实现；

at2p, bt2p, ct2p, dt2p：传递函数 $T_{21}^{\perp}$的状态空间实现；

kx, x, ky, y, f, h：连续 LQR 综合的计算结果，对应的 MATLAB 程序为：

```
[kx,x] = lqrc(A,B2,C1'*C1,D12'*D12,C1'*D12);
[ky,y] = lqrc(A',C2',B1*B1',D21*D21',B1*D21');
f = -kx;
h = -ky';
```

函数 youla 计算结果将用于基于 Hankel 最优逼近的 H_∞综合。

4.8 示　　例

鲁棒控制工具箱提供了丰富的示例程序，它们在表 4-11 中列出。

表4-11　示例命令列表说明

示 例 名	说　明
accdemo	突发基准问题
dintdemo	H_∞设计
hinfdemo	H_∞& H^2设计示例
ltrdemo	LQG/LTR 设计示例
mudemo, mudemo1	μ 综合示例
mrdemo	鲁棒模型降阶示例
rctdemo	鲁棒控制工具箱示例主菜单

用户可以通过 MATLAB 的命令行中直接键入上述命令进入演示程序，也可以通过系统的 demo 窗口选择鲁棒控制工具箱进入演示画面。所有的命令行演示均可以通过 rctdemo 获得，如图4-18 所示。

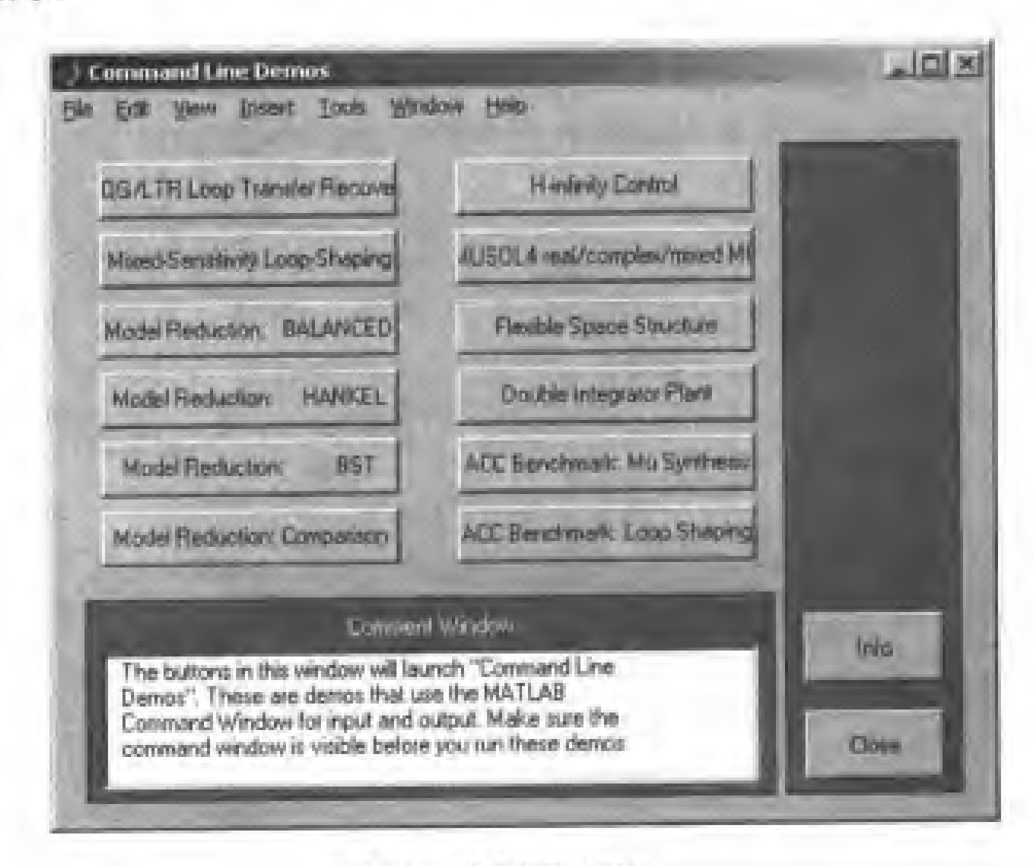


图4-18　命令行演示窗口

另外 MATLAB 的 demo 中还提供了许多演示，如图4-19 所示。

下面以 H_infinity Controller 示例为代表，讲述示例的使用方法。

选中 H_infinity Controller，选择运行，可以得到 H_∞设计的系统方框图，如图4-20 所示。

系统在装载该演示时，会自动加载演示的数据，MALTAB 窗口中会显示如下信息：

loading H_infinity data...

Done

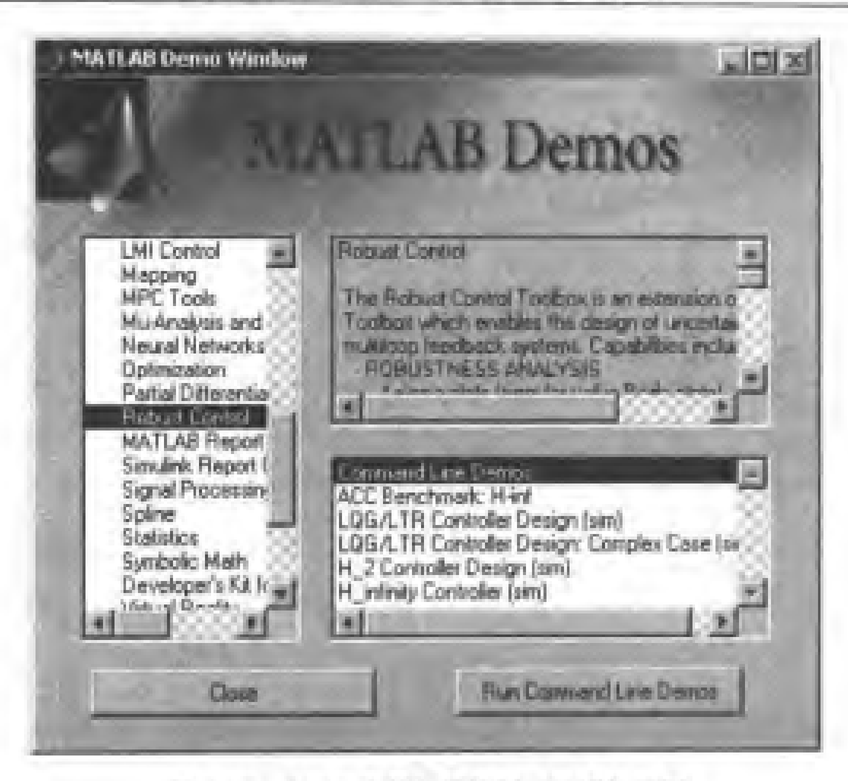


图 4-19 demo 中的鲁棒控制工具箱的示例

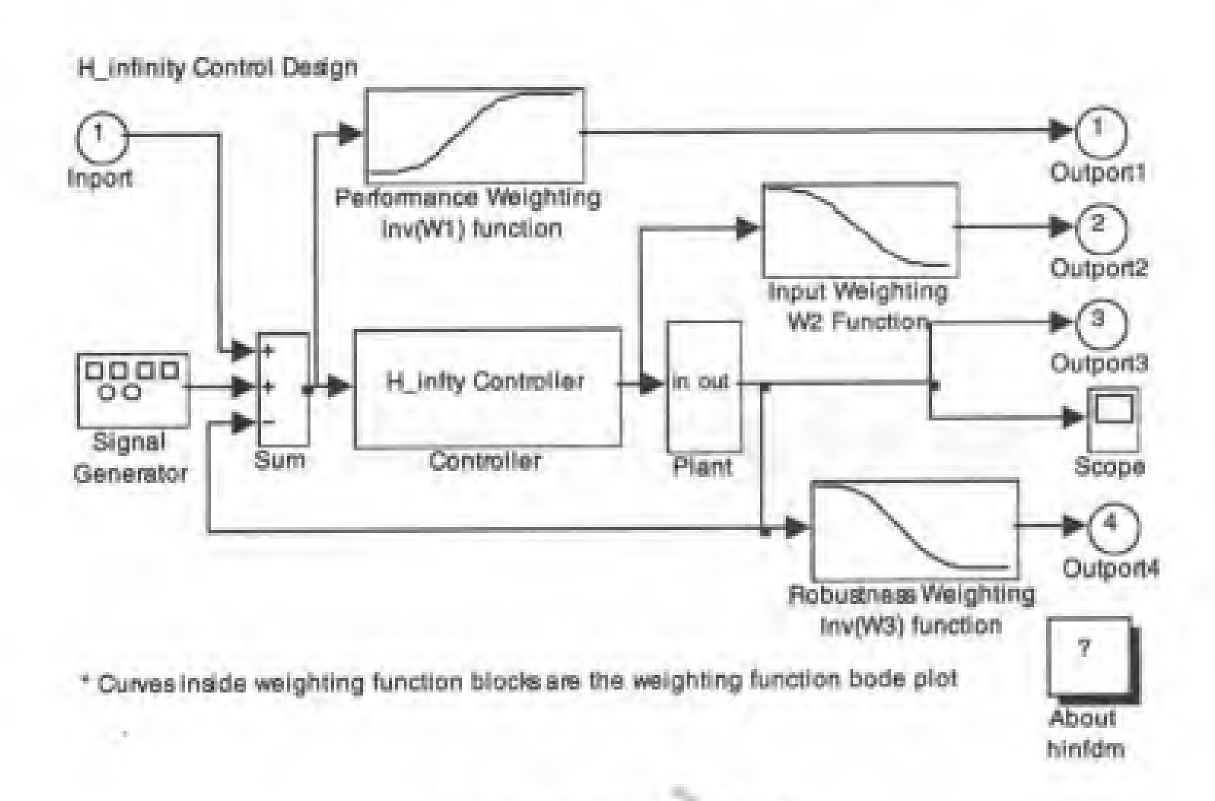


图 4-20 H_∞设计的系统方框图

如果由于某种原因加载数据失败，可以在系统方框图下方选择 Re-Load Data。同时 MATLAB 还打开观察系统响应曲线的窗口如图 4-21 所示。

选择 ▤ (parameters) 按钮，可以打开观察窗的参数设置窗口，如图 4-22 所示。





图 4-21　系统响应曲线窗口

该窗口有两个选项卡，General 为一般图形坐标参数的设置，Data history 为如何处理历史数据，如图 4-23 所示，当选中 Save Data to workspace 时，MATLAB 会自动将模拟的数据保存在工作空间中。

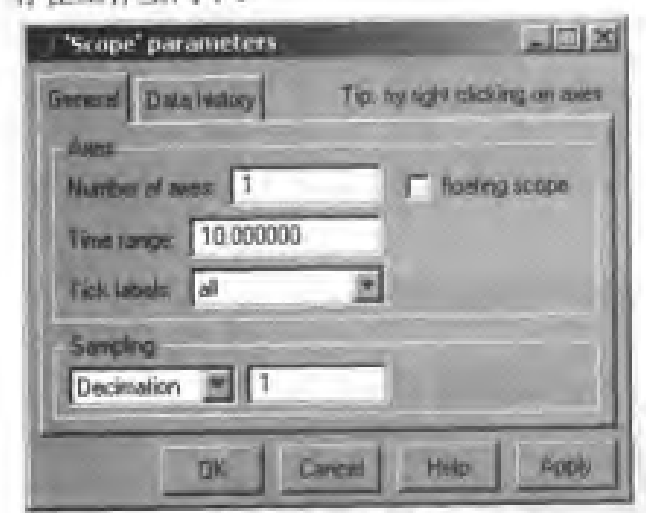


图 4-22　观察窗参数设置窗口



图 4-23　历史数据选项卡

在完成这两项设置后可以开始演示，在设计完成时，观察窗口中曲线如图 4-24 所示。

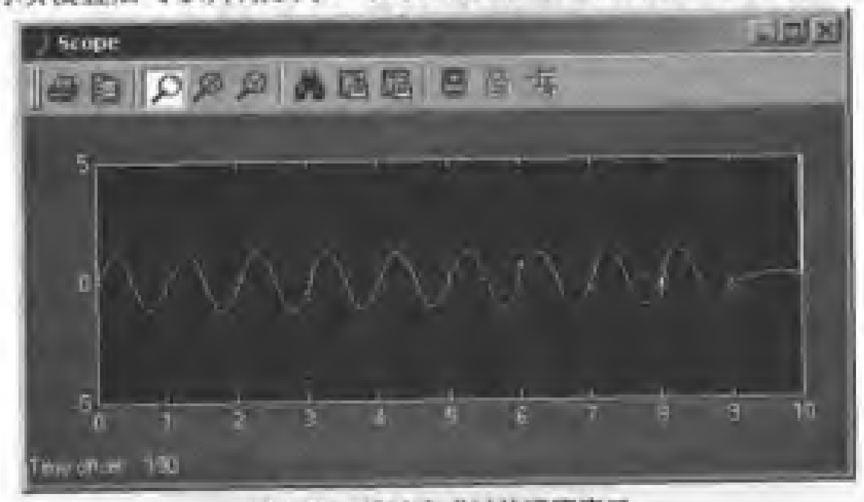


图 4-24　设计完成时的观察窗口

如果选种 按钮，可选择显示多种响应曲线。可以单击 进入绘制信号选择设置窗口，如图 4-25 所示。

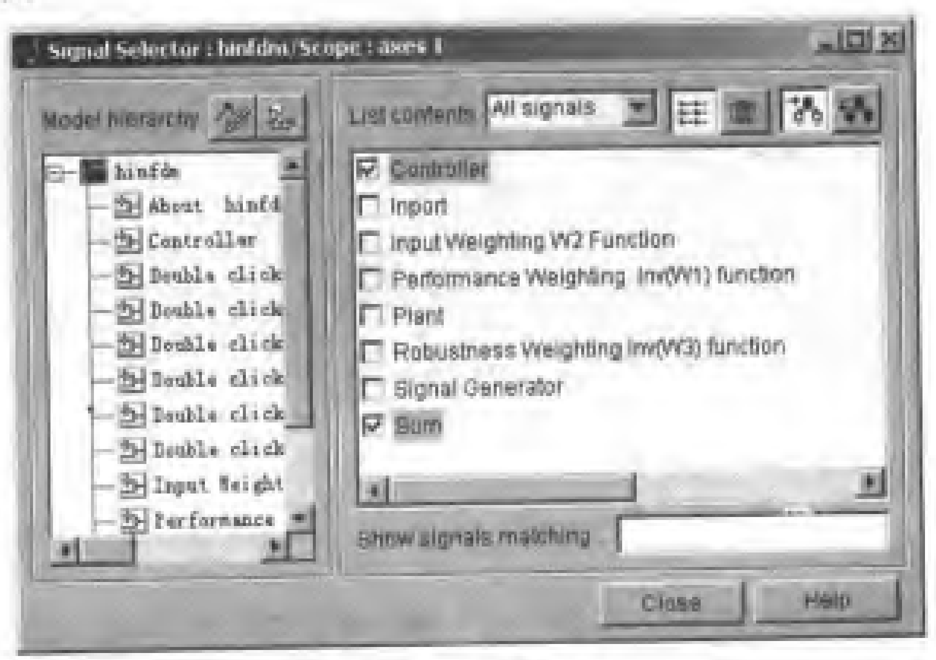


图 4-25 信号选择窗口

选中的信号将在 Scope 窗口中显示，当如图 4-25 中选中 Controller 和 Sum 后，运行演示，将得到的 Scope 窗口如图 4-26 所示。

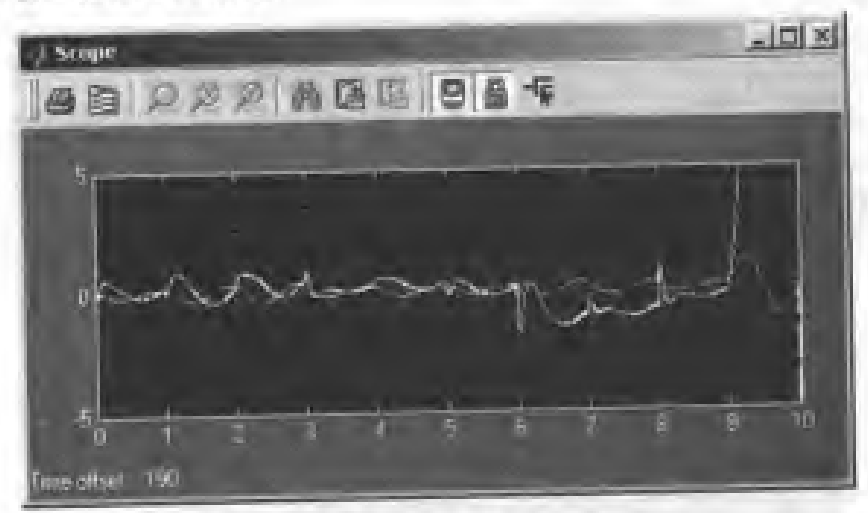


图 4-26 选中两个信号的 Scope 窗口图

第5章 模型预测控制工具箱

在20世纪60年代形成并发展起来的基于状态空间方法的现代控制论，具有最优的性能指标和精确的理论设计方法，在空间技术等领域取得了很大的成功。但在工程的过程控制中却很难得到应用，原因在于现代控制理论的基础就是精确的数学模型，如果模型不准确，则控制系统的特性将大大降低，而工业上又很难得到精确的数学模型。

为了克服现代控制论的缺点，20世纪70年代以来人们从工业过程出发，寻找对模型要求不高、而又能实现的最佳控制方法，最初是由美国和法国的几家石油公司提出的。模型预测控制就是在这种情况下发展起来的一种新型的计算机控制算法。该算法直接应用于工业，并在使用中不断发展并成熟。

模型预测需要一个描述系统动力行为的基础模型，能够根据被控制对象的历史信息和未来输入来预测系统的未来响应。模型预测可以为脉冲响应、阶跃响应等非参数模型，也可以为微分方程、差分方程等参数模型。

5.1 系统模型辨识函数

为了进行模型预测控制器设计，需要对系统进行辨识，MATLAB提供的系统辨识函数见表5-1。

表5-1 系统模型辨识函数列表说明

函数名	函数功能说明
autosc	矩阵或向量的自动归一化
imp2step	由MISO脉冲响应模型生成MIMO阶跃响应模型
mlr	利用多变量线性回归计算MISO脉冲响应模型
plsr	利用部分最小二乘方回归方法计算MISO脉冲响应模型
rescal	由归一化的数据生成原数据
scal	根据指定的均值和标准差归一化矩阵
validmod	利用新的数据检验MISO脉冲响应模型
wrtreg	生成用于线性回归计算的数据矩阵

5.1.1 数据向量或矩阵的归一化

1. autosc

功能：矩阵或向量的自动归一化。

语法：[ax,mx,stdx] = autosc(x)

说明：输入参数：

x：数据向量或矩阵。

输出参数：

ax：归一化后的向量或矩阵。若输出参数 x 为矩阵，则对其每一列进行归一化计算。

mx：均值向量。当输入参数 x 为矩阵时，mx 为 x 各列向量的均值构成的向量。

stdx：标准差向量。当输入参数 x 为矩阵时，stdx 为 x 的各列向量的标准差构成的向量。

【例 1】

```
x=[1 1 1 2];
[ax,mx,stdx] = autosc(x);
```

MATLAB 返回：

```
ax =
-0.5000    -0.5000    -0.5000    1.5000
mx =
            1.2500
stdx =
0.5000
```

2．scal

功能：根据指定的均值和标准差归一化矩阵。

语法：sx = scal(x,mx)

sx = scal(x,mx,stdx)

说明：输入参数：

x：数据向量或矩阵；

mx：均值向量。当输入参数 x 为矩阵时，mx 为 x 各列向量的均值构成的向量。

stdx：标准差向量。当输入参数 x 为矩阵时，stdx 用于指定对 x 的各列向量的进行归一化的标准差。

输出参数 sx 为归一化以后的向量或矩阵。若输入参数 x 为矩阵时，则对每一列进行归一化计算。

【例 2】

```
x=[1 2 ; 2 1];
s=scal(x,[1 1], [0.5 0.5])
```

MATLAB 返回：

```
s =
      0     2
      2     0
```

3．rescal

功能：由归一化的数据生成原数据。

语法：rx = rescal(x,mx)

rx = rescal(x,mx,stdx)



说明：输入参数：

x：数据向量或矩阵；

mx：用于反归一化的均值向量；

stdx：用于反归一化的标准差向量；

输出参数定义为：

rx：反归一化得到的向量或矩阵。

【例 3】

```
x=[1  2  ;  2   1];
s=rescal(x,[1 1], [0.5 0.5])
```

MATLAB 返回：

```
s =
     1.5000    2.0000
     2.0000    1.5000
```

5.1.2 基于线性回归方法的脉冲响应模型辨识

1. mlr

功能：利用多变量线性回归计算 MISO 脉冲响应模型

语法：[theta, yres] = mlr(xreg,yreg,ninput)

[theta, yres] = mlr(xreg,yreg,ninput,plotopt,wtheta, wdeltheta)

说明：输入参数：

xreg：预处理后的输入数据矩阵；

yreg：预处理后的输出数据矩阵；

ninput：输入变量的个数；

plotopt：绘图选项：

- plotopt=0：缺省值，不绘制图形；
- plotopt=1：绘制实际输出和预测输出曲线；
- plotopt=2：绘制实际输出、预测输出以及输出误差；

wtheta, wdeltheta：最小二乘法的加权向量，缺省为 0。

输出参数：

theta：脉冲响应模型的系数矩阵。该矩阵的每一列对应一个输入到输出的脉冲响应模型系数向量；

yres：预测误差向量。

【例 1】 考虑如下的两输入单输出系统：

$$y(s)=\begin{bmatrix}\dfrac{5.72\mathrm{e}^{-14s}}{60s+1} & \dfrac{1.52\mathrm{e}^{-15s}}{25s+1}\end{bmatrix}\begin{bmatrix}u_1(s)\\ u_2(s)\end{bmatrix}$$

采样周期为 7s，其输入输出数据在数据文件 mlrdat.dat 中；

系统实际输出和模型预测输出及预测误差曲线如图 5-1 所示。

```
load mlrdat;
```

```
[ax,mx,stdx] = autosc(x);
mx = [0,0];
sx = scal(x,mx,stdx);
n = 35;
[xreg,yreg] = wrtreg(sx,y,n);
ninput = 2;
plotopt = 2;
[theta,yres] = mlr(xreg,yreg,ninput,plotopt);
```

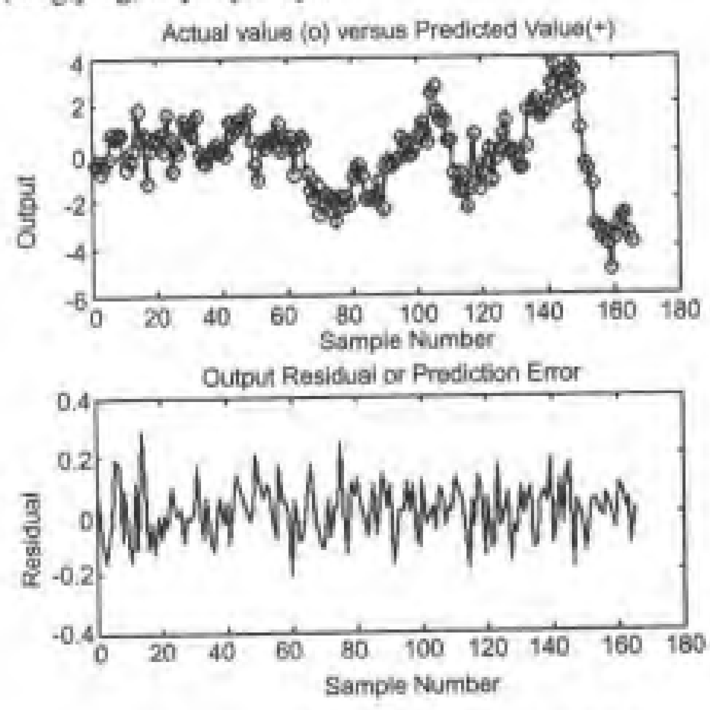


图 5-1　脉冲响应模型预测输入与预测误差曲线

2．plsr

功能：利用部分最小二乘方（PLS）方法计算 MISO 脉冲响应模型。

语法：[theta,w,cw,ssqdif,yres] = plsr(xreg,yreg,ninput,lv)

[theta,w,cw,ssqdif,yres] = plsr(xreg,yreg,ninput,lv, plotopt)

说明：输入参数定义为：

xreg：预处理后的输入数据矩阵；

yreg：预处理后的输出数据矩阵；

ninput：输入变量的个数；

plotopt：绘图选项：

- plotopt=0：缺省值，不绘制图形；
- plotopt=1：绘制实际输出和预测输出曲线；
- plotopt=2：绘制实际输出。

Lv：用于指定最大化输入输出的协方差的方向的数目。

输出参数：

theta：脉冲响应模型的系数矩阵；

w,cw,ssqdif：部分最小二乘方的有关计算结果；

yres：预测输出误差。

【例 2】 考虑如下的两输入单输出的系统：

$$y(s)=\begin{bmatrix}\dfrac{5.72e^{-14s}}{60s+1} & \dfrac{1.52e^{-15s}}{25s+1}\end{bmatrix}\begin{bmatrix}u_1(s)\\u_2(s)\end{bmatrix}$$

输入输出数据存储在数据文件 plsrdat.mat 中，模型预测输入和预测误差如图 5-2 所示。

```
load plsrdat;
n = 30;
[xreg,yreg] = wrtreg(x,y,n);
ninput = 2;
lv = 10;
plotopt = 2;
theta = plsr(xreg,yreg,ninput,lv,plotopt);
```

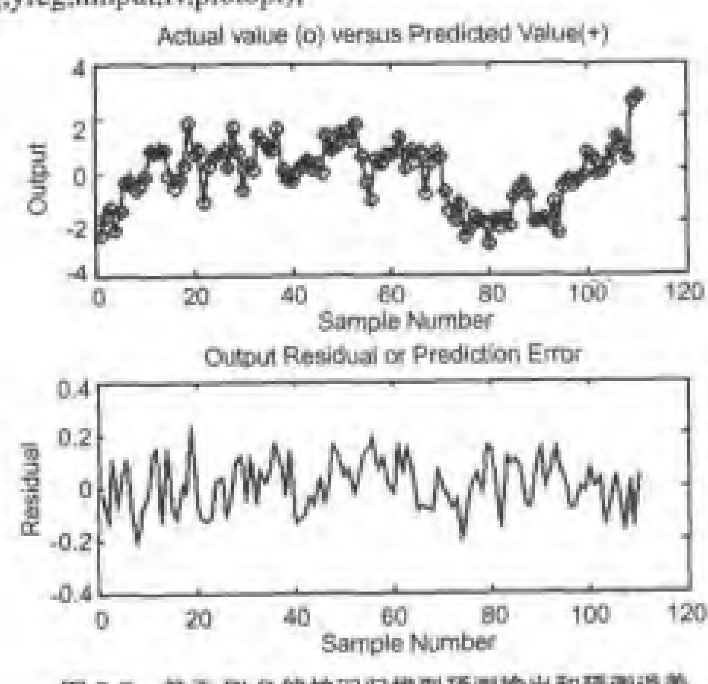


图 5-2　基于 PLS 线性回归模型预测输出和预测误差

3．wrtreg

功能：生成用于线性回归计算的数据矩阵。

语法：[xreg,yreg] = wrtreg(x,y,n)

说明：输入参数：

x：输入数据矩阵；

y：输出数据向量；

n：脉冲响应模型系数的个数。

输出参数：

xreg：预处理后的输入数据矩阵；

yreg：预处理后的输出数据矩阵。

5.1.3 脉冲响应模型转换为阶跃响应模型

1．imp2step

功能：脉冲响应模型转换为阶跃响应模型。

语法：plant = imp2step(delt,nout,theta1,theta2, ...,theta25)

说明：输入参数定义：

delt：脉冲响应模型采用的采样周期；

nout：用于指定输出的稳定性。对于稳定系统，nout 等于输出的个数；对于一个具有多个积分输出的系统，nout 为一个长度等于输出个数的向量，该向量对应的积分输出的分量为 0，其余的分量为 1。

theta1,theta2, ...,theta25：脉冲响应系数矩阵，维数为(n*ny+ny+2) ×nu，其中 n 为对应每个输入的系数个数，nu 和 ny 分别为输入和输出变量的个数。

【例 1】 将脉冲响应模型转换为阶跃响应模型，并绘制阶跃响应曲线，如图 5-3 所示。

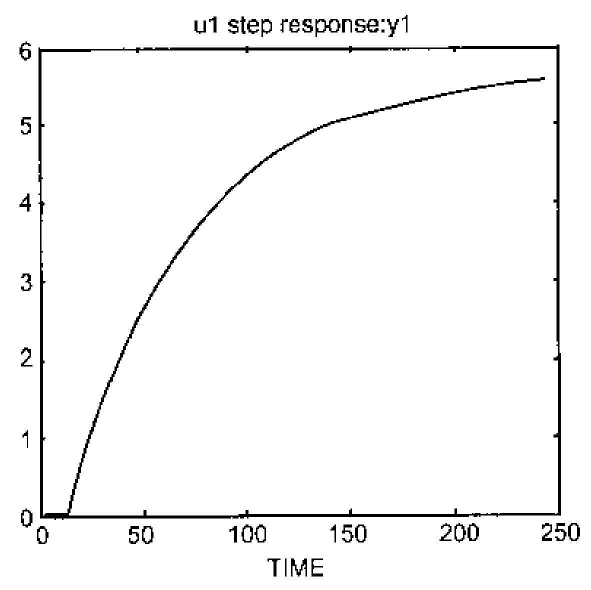


图 5-3 系统阶跃响应预测曲线

```
load mlrdat;
[ax,mx,stdx] = autosc(x);
mx = [0,0];
sx = scal(x,mx,stdx);
n = 35;
[xreg,yreg] = wrtreg(sx,y,n);
ninput = 2;
plotopt = 2;
[theta,yres] = mlr(xreg,yreg,ninput,plotopt);
theta = scal(theta,mx,stdx);
nout = 1;
delt = 7;
```




```
model = imp2step(delt,nout,theta);
plotstep(model)
```

5.1.4 模型的校验

1. validmod

功能：利用新的数据对模型进行校验

语法：yres = validmod(xreg,yreg,theta)

yres = validmod(xreg,yreg,theta,plotopt)

说明：输入参数定义：

xreg：经过预处理的输入校验数据；

yreg：经过预处理的输出校验数据；

theta：脉冲响应模型的系数矩阵；

plotopt：绘图选项：

- plotopt=0：缺省值，不绘制图形；
- plotopt=1：绘制实际输出和预测输出曲线；
- plotopt=2：绘制实际输出。

输出参数定义：

yres：输出预测误差。

5.2 系统矩阵信息及绘图函数

MATLAB 模型预测工具箱提供的一系列函数绘制有关系统的特性，见表 5-2。

表 5-2 系统矩阵信息及绘图函数列表说明

函数名	函数功能说明
mpcinfo	返回矩阵的类型和属性
plotall	在一个图形窗口中绘制系统仿真的输入输出曲线
plotfrsp	绘制系统频率响应的波特图
ploteach	在多个图形窗口中分别绘制系统的输入输出仿真曲线
plotstep	绘制系统阶跃响应模型的曲线

1. mpcinfo

功能：返回矩阵的类型和属性。

语法：mpcinfo(mat)

说明：输入参数 mat 为系统模型矩阵，函数返回该矩阵的类型和尺度。

如果系统为 MPC 阶跃模型，输出包含采样周期，输入变量和输出变量的数目，阶跃响应系数的数目；

假如系统为 MPC mod 模型，输出包含采样周期，状态数目，应用输入变量数目，测量扰动、未测量扰动数目，测量输出和未测量输出。

【例 1】

(1) MPC 阶跃响应模型

```
phi=diag([0.3, 0.7, -0.7]);
gam=eye(3);
c=[1 0 0;   0 0 1;   0 1 1;   0 1 0];
d=[1 0 0; zeros(3,3)];
nstep=4; delt=1.5;
plant=mod2step(ss2mod(phi,gam,c,d,delt),(nstep-1)*delt);
mpcinfo(plant)
```

系统返回矩阵信息：

```
This is a matrix in MPC Step format.
sampling time = 1.5
number of inputs = 3
number of outputs = 4
number of step response coefficients = 3
All outputs are stable.
```

(2) MPC mod 模型

```
[phiu,gamu,cu,du]=tf2ss(0.7,[1 -0.9]);
[phid,gamd,cd,dd]=tf2ss(-1.5,[1 -0.85]);
[phiw,gamw,cw,dw]=tf2ss(1,[1 0.6]);
[phi,gam,c,d]=mpcparal(phiu,gamu,cu,du,phid,gamd,cd,dd);
[phi,gam,c,d]=mpcparal(phi,gam,c,d,phiw,gamw,cw,dw);
delt=2;
minfo=[delt 3 1 1 1 1 0];
pmod=ss2mod(phi,gam,c,d,minfo):
mpcinfo(pmod)
```

系统返回矩阵信息：

```
This is a matrix in MPC Mod format.
     minfo = [2 3 1 1 1 1 0 ]
sampling time = 2
number of states = 3
number of manipulated variable inputs = 1
number of measured disturbances = 1
number of unmeasured disturbances = 1
number of measured outputs = 1
number of unmeasured outputs = 0
```

2. plotall

功能：绘制系统仿真的输入输出曲线。

语法：plotall(y,u)

plotall(y,u,t)

说明：输入参数定义：y,u 为系统的输入输出变量，其中控制量 u 在绘图之前转换为接替型的连续数据变量；t 为采样周期。

3. plotfrsp

功能：绘制系统的频率响应波特图。

语法：plotfrsp(vmat)

plotfrsp(vmat,out,in)

说明：输入参数定义：

vmat：系统频率响应矩阵；

out：指定输出变量；

in：指定输入变量。

【例 2】 绘制系统的频率响应波特图，如图 5-4 所示。

```
mpc=poly2tfd(3,[6 1],0,4);
mod=tfd2mod(0.1 , 1,mpc);
frsp=mod2frsp(mod,[-2,3,50],1,1,0);
plotfrsp(frsp)
```

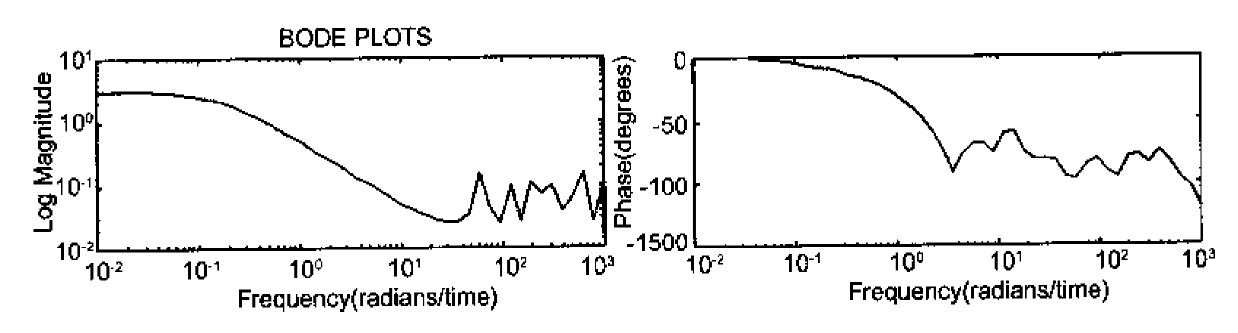


图 5-4　系统频率响应曲线

4. ploteach

功能：在多个窗口绘制系统仿真的输入输出曲线。

语法：ploteach(y)

ploteach(y,u)

ploteach([ ],u)

ploteach(y,[ ],t)

ploteach([ ],u,t)

ploteach(y,u,t)

说明：输入参数定义：y,u 为系统的输入输出变量，其中控制量 u 在绘图之前转换为接替型的连续数据变量；t 为采样周期。

5. plotstep

功能：绘制系统阶跃响应模型曲线。

语法：plotstep(plant)

plotstep(plant,opt)

说明：输入参数 plant 为对象阶跃响应模型；opt 为指定输出变量。

【例3】 绘制系统阶跃响应模型曲线，如图 5-5 所示。

```
g=poly2tfd(3,[5 1],0,4);
step=tfd2step(30,2,1,g);
plotstep(step)
```

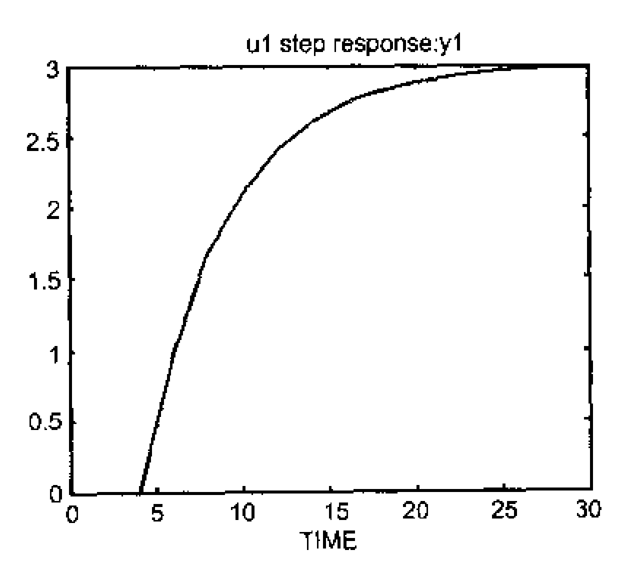


图 5-5 系统阶跃响应曲线

5.3 模型转换函数

在 MATLAB 模型预测控制工具箱中提供了许多函数可以实现各种模型间的转换。表 5-3 列出了这些函数。

表 5-3 模型转换函数列表说明

函数名	函数功能说明
c2dmp	将连续时间状态空间模型转换为离散时间状态空间模型
cp2dp	将连续传递函数模型转换为离散传递函数模型
d2cmp	将离散时间状态空间模型转换为连续时间状态空间模型
mod2mod	改变 MPC mod 模型的采样周期
mod2ss	将 MPC mod 模型转换为状态空间模型
mod2step	将 MPC mod 模型转换为阶跃响应模型
poly2tfd	将多项式的传递函数模型转换为 MPC 传递函数模型
ss2mod	将状态空间模型转换为 MPC mod 模型
ss2step	将状态空间模型转换为阶跃响应模型
ss2tf2	将状态空间模型转换为传递函数模型
tf2ssm	将传递函数模型转换为状态空间模型
tfd2mod	将 MPC 传递函数模型转换为 MPC mod 模型
tfd2step	将 MPC 传递函数模型转换为阶跃响应模型
th2mod	将 Theta 格式模型转换为 MPC mod 模型

注：c2dmp、cp2dp、ss2tf2 和 tf2ssm 分别等价于控制系统工具箱中的函数：c2d、d2c、ss2tf 和 tf2ss，所以这里就不列出它们的详细说明。可以参考第 4 章控制系统工具箱的相应函数的具体说明。

1．cp2dp

功能：将连续传递函数模型转换为离散传递函数模型。

语法：[numd,dend] = cp2dp(num,den,delt)

[numd,dend] = cp2dp(num,den,delt,delay)

说明：输入参数定义：

num 和 den 分别为连续时间传递函数的分子和分母系数向量；

delt 为采样周期；

可选参数 delay 为系统的时延，如果省略则认为无时延。

输出参数定义：

numd 和 dend 分别为返回的离散时间传递函数的分子和分母系数向量。

2．mod2mod

功能：改变 MPC mod 模型的采样周期。

语法：newmod = mod2mod(oldmod,delt2)

说明：输入参数定义：

oldmod：原有离散系统的 MPC mod 模型；

delt2：指定的新的采样周期。

输出参数 newmod 为使用新采样周期后的系统 MPCmod 模型。

【例 4】 考虑一个 3 阶系统，并用 0.5s 采样，绘制阶跃响应曲线，如图 5-6 所示。

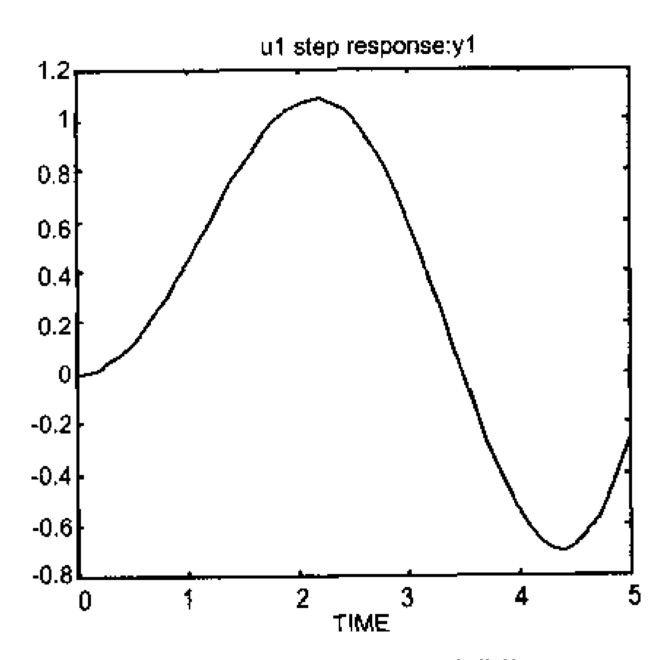


图 5-6　三阶对象阶跃响应曲线

```
num=[1 2];
den=[1 2 1 5];
tf=poly2tfd(num,den,0,0);
```

```
m1=tfd2mod(0.1,1,tf);
%将系统重新采样
m2=mod2mod(m1,0.5);
p=mod2step(m2,5,0.1);
plotstep(p)
```

3. mod2ss

功能：MPC mod 模型转换为状态空间模型。

语法：[phi,gam,c,d] = mod2ss(mod)

[phi,gam,c,d,minfo] = mod2ss(mod)

说明：输入参数 mod 为系统的 MPC mod 模型格式。

输出参数 phi,gam,c,d 为状态空间矩阵；minfo 为构成 MPC mod 模型的其他描述信息。

4. mod2step

功能：MPC mod 模型转换为阶跃响应模型。

语法：plant = mod2step(mod,tfinal)

[plant,dplant] = mod2step(mod,tfinal,delt2,nout)

说明：输入参数定义 mod 为系统的 MPC mod 模型；

tfinal 为系统阶跃响应模型的截断时间；

delt2 为采样周期；

nout 为输入稳定性向量。对于稳定系统，nout 等于输出的个数，对于具有一个或多个积分输出系统，nout 为一个长度等于输出个数的向量，该向量对应积分输出的分量为 0，其余的分量为 1。

输出参数 plant 和 dplant 均为 MPC 阶跃响应格式的矩阵，

plant 为对象在受控变量作用下的阶跃响应系数，矩阵为 n*ny + ny + 2 行，nu 列；

dplant 为对象在扰动作用下的阶跃响应矩阵 n*ny + ny + 2 行，nd + nw 列。

其中 n = round (tfinal/delt2)，nu 和 ny 分别为输入和输出变量的个数。

例子参见函数 mod2mod 例子。

5. poly2tfd

功能：将通用的传递函数模型转换为 MPC 传递函数模型。

语法：g = poly2tfd(num,den)

g = poly2tfd(num,den,delt,delay)

说明：输入参数定义：

num,den 传递函数的分子分母多项式的系数向量；

delt 为系统采样周期；

delay 系统纯时延，如果系统为离散时间，则纯时延为采样周期的整数倍。

输出参数 g 为对象 MPC 传递函数模型。

【例 5】 考虑连续时间传递函数模型转换为 MPC 传递函数模型：

```
g=poly2tfd(0.5*[3 -1],[5 2 1]);
```

假如系统具有 2.5s 延迟，则有：

```
g=poly2tfd(0.5*[3 -1],[5 2 1],0,2.5);
```




6．ss2mod

功能：状态空间模型转换为 MPC mod 模型。

语法：pmod = ss2mod(phi,gam,c,d)

　　　pmod = ss2mod(phi,gam,c,d,minfo)

说明：输入参数 phi,gam,c,d 为状态空间矩阵；

minfo 为构成 MPC mod 模型的描述性信息，为长度 7 的向量，其定义见表 5-4。

表 5-4　minfo 取值意义说明

minfo(1)=T	系统采样周期
minfo(2)=n	系统阶次
minfo(3)=nu	受控输入的个数
minfo(4)=nd	测量扰动的数目
minfo(5)=nw	未测量扰动的数目
minfo(6)=nym	测量输出的数目
minfo(7)=nyu	未测量输出的数目

【例 6】

```
num=[2 1 2];
den=[4 3 2 1];
[phiu,gamu,cu,du]=tf2ss(num,den);
mod = ss2mod(phiu,gamu,cu,du)
```

MALTAB 返回：

```
mod =
    1.0000    3.0000    1.0000    0          0          1.0000    0
    NaN       -0.7500   -0.5000   -0.2500    1.0000     0         0
    0         1.0000    0         0          0          0         0
    0         0         1.0000    0          0          0         0
    0         0.5000    0.2500    0.5000     0          0         0
```

7．ss2step

功能：状态空间模型转换为阶跃响应模型。

语法：plant = ss2step(phi,gam,c,d,tfinal)

　　　plant = ss2step(phi,gam,c,d,tfinal,delt1,delt2,nout)

说明：输入参数 phi,gam,c,d 为系统状态空间矩阵；

tfinal：为阶跃响应截断时间；

delt1：为系统采样周期；

delt2：阶跃响应模型采样周期；

nout：输出稳定性向量，参见 mod2step 说明。

输出参数 plant 为对象阶跃响应对应的矩阵。

【例 7】 考虑一个 3 输入 4 输出的系统，转换为阶跃响应模型后，绘制系统阶跃响应曲线如图 5-7。

```
phi=diag([0.3, 0.7, -0.7]);
gam=eye(3);
c=[1 0 0; 0 0 1; 0 1 1; 0 1 0];
d=[1 0 0; zeros(3,3)];
delt1=1.5; tfinal=3*1.5;
plant=ss2step(phi,gam,c,d,tfinal,delt1);
plotstep(plant)
```

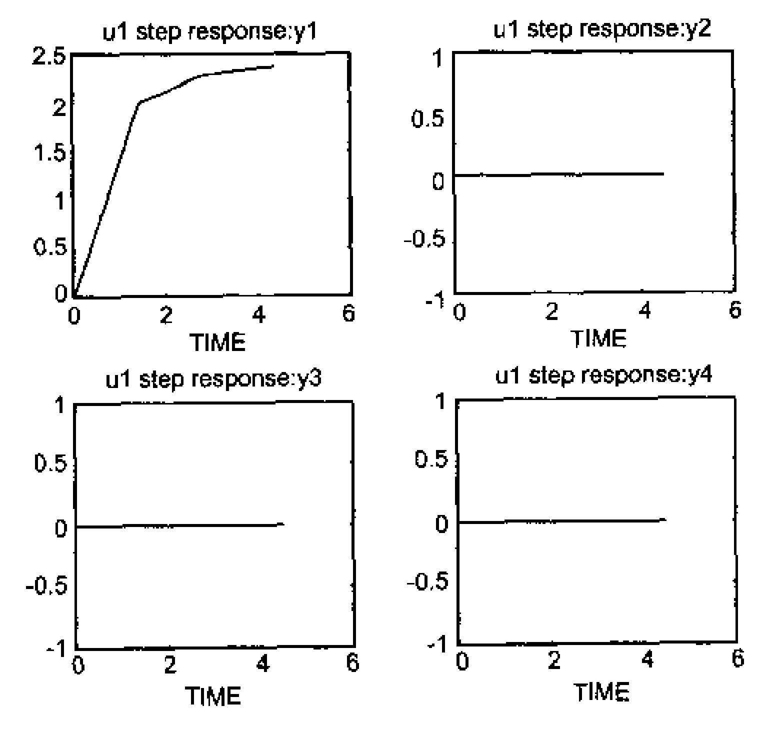


图 5-7　系统阶跃响应曲线

8. tfd2mod

功能：MPC 传递函数模型转换为 MPC mod 模型。

语法：model = tfd2mod(delt2,ny,g1,g2,g3,...,g25)

说明：输入参数说明：

delt2：采样周期；

ny：系统输出的个数；

g1,g2,g3,...,g25：SISO 传递函数。

输出 model 为 MPC mod 模型。

考虑如下传递函数

$$G(s)=\frac{b_0s^n+b_1s^{n-1}+\cdots+b_n}{a_0s^n+a_1s^{n-1}+\cdots+a_n}$$

或者离散时的情形

$$G(z)=\frac{b_0+b_1z^{-1}+\cdots+b_nz^{-n}}{a_0+a_1z^{-1}+\cdots+a_nz^{-n}}$$

MPC 传递函数格式矩阵有 3 行：

第一行：分子系数从 b_0 到 b_n

第二行：分母系数从 a_0 到 a_n

第三行：第一列为采样周期，如果为连续时间系统则为 0，否则为正数。第二列为系统纯时延，对连续时间系统为时间值，对离散时间系统该值为采样周期的倍数。

【例 8】 考虑如下的线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \begin{bmatrix} \dfrac{3.8e^{-8s}}{14.9s+1} \\ \dfrac{4.9e^{-3s}}{13.2s+1} \end{bmatrix} w(s)$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=2;
ny=2;
umod=tfd2mod(delt,ny,g11,g21,g12,g22);
gw1=poly2tfd(3.8,[14.9 1],0,8);
gw2=poly2tfd(4.9,[13.2 1],0,3);
wmod=tfd2mod(delt,ny,gw1,gw2);
pmod=addumd(umod,wmod);
```

9. tfd2step

功能：传递函数模型转换为阶跃响应模型。

语法：plant = tfd2step(tfinal,delt2,nout,g1)

plant = tfd2step(tfinal,delt2,nout,g1,...,g25)

说明：输入参数说明：

tfinal：阶跃响应的阶段时间；

delt2：采样周期；

ny：系统输出的个数；

g1,g2,g3,...,g25：SISO 传递函数。

【例 9】 考虑如下多变量系统传递函数模型转换为阶跃响应模型：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

系统的阶跃响应图如 6-8 所示。

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
```

```
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=2;
ny=2;
gw1=poly2tfd(3.8,[14.9 1],0,8);
gw2=poly2tfd(4.9,[13.2 1],0,3);
tfinal=90;
plant=tfd2step(tfinal,delt,ny,g11,g21,g12,g22,gw1,gw2);
plotstep(plant)
```

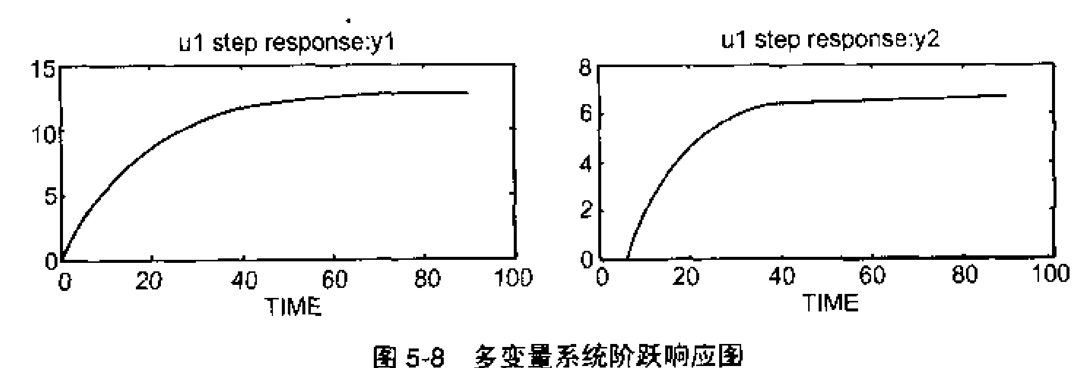


图 5-8　多变量系统阶跃响应图

10. th2mod

功能：Theta 格式模型转换为 MPC mod 模型。

语法：umod = th2mod(th)

　　[umod,emod] = th2mod(th1,th2,...,thN)

说明：输入参数为一个或多个 Theta 格式模型。

输出参数 umod 为系统的测量输入对输出影响的 MPC mod 模型；

emod 系统的噪声输入对输出影响的 MPC mod 模型。

5.4　模型建立和连接函数

MALTAB 模型预测工具箱提供了将模型进行互连的函数，用来建立复杂的系统模型，见表 5-5。

表 5-5　模型建立和连接函数列表说明

函数名	函数功能说明
addmd	向对象添加一个或多个测量扰动
addmod	连接两个系统，使其中一个系统的输出迭加到另一个系统的输入
addumd	向对象添加一个或多个未测量扰动
appmod	将两个系统模型增广成具有无连接平行的结构的系统
paramod	并联两个模型系统
sermod	将两个模型串联

1. addmd

功能：以 MPC mod 的格式向对象输出添加测量扰动。

语法：model = addmd(pmod,dmod)

说明：输入参数 pmod 为对象的 MPC mod 模型；

dmod：测量扰动的 MPC mod 模型。

输出参数 model 为加入了测量扰动后的 MPC mod 模型。

2. addmod

功能：形成闭环系统模型。

语法：pmod = addmod(mod1,mod2)

说明：输入参数 mod1 和 mod2 分别为两个 MPC mod 系统，mod2 的输出迭加到 mod1 的输入。

输出 pmod 为闭环系统模型。

【例 1】 将两个 MPC mod 系统连接成闭环模型。

```
num=[1 2];
den=[1 2 1 5];
num1=[3 2];
den1=[2 1 5];
tf1=poly2tfd(num,den,0,0);
tf2=poly2tfd(num1,den1,0,0);
m1=tfd2mod(0.1,1,tf1);
m2=tfd2mod(0.1,1,tf2);
p=addmod(m1,m2)
```

3. addumd

功能：向对象输出添加未测量扰动信号。

语法：model = addumd(pmod,dmod)

说明：输入参数 pmod 为对象的 MPC mod 模型；

dmod：未测量扰动的 MPC mod 模型。

输出参数 model 为加入了未测量扰动后的 MPC mod 模型。

4. appmod

功能：计算两个 MPC mod 模型构成的增广系统模型。

语法：pmod = appmod(mod1,mod2)

说明：输入参数 mod1,mod2 为两个 MPC mod 模型。

输出 pmod 为增广系统的 MPC mod 模型。

5. paramod

功能：计算两个系统的并联模型。

语法：pmod = paramod(mod1,mod2)

说明：输入参数 mod1 和 mod2 为两个并联子系统的 MPC mod 模型。

输出参数 pmod 为并联后的 MPC mod 模型。

该系统输出为两个子系统输出之和。系统的输入按类型重新分组，如图 5-9 所示。

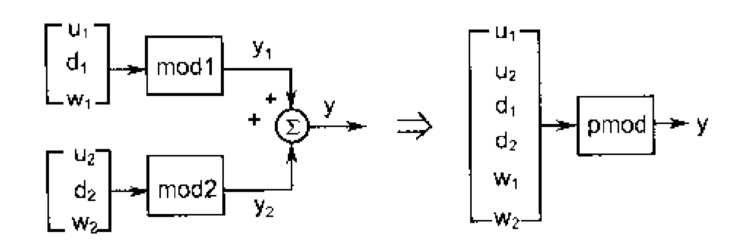


图 5-9　并联系统示意图

6. sermod

功能：计算两个系统的串联模型实现。

语法：pmod = sermod(mod1,mod2)

说明：输入参数 mod1 和 mod2 为两个串联的子系统 MPC mod 模型。

输出 pmod 为串联后的 MPC mod 模型。

在该系统中，mod1 的测量输出作为 mod2 的控制输入，两个子系统的扰动信号仍然作为串联系统的扰动信号，如图 5-10 所示。

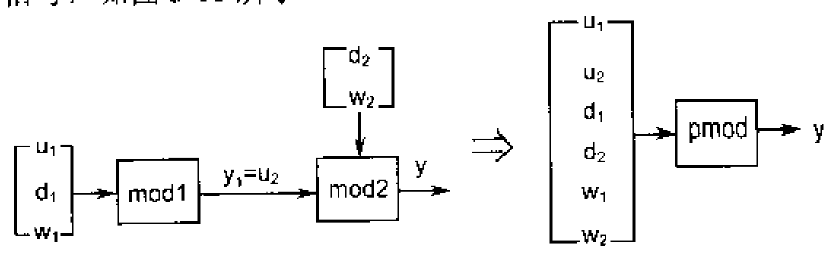


图 5-10　串联系统示意图

5.5　控制器设计与仿真

5.5.1　基于 MPC 阶跃响应的控制器设计与仿真

基于 MPC 阶跃响应的控制器设计也称为动态矩阵法，MATLAB 模型预测控制工具箱提供的控制器设计与仿真函数见表 5-6。

表 5-6　基于 MPC 阶跃响应的控制器设计与仿真函数

函 数 名	函 数 功 能 说 明
cmpc	输入输出受限的模型预测控制器设计与仿真
mpccl	计算模型预测控制系统的闭环模型
mpccon	输入输出不受限的模型预测控制器设计
mpcsim	输入输出不受限的模型预测闭环控制系统仿真
nlcmpc	Simulink 块 nlcmpc 对应的 S 函数
nlmpcsim	Simulink 块 nlmpcsim 对应的 S 函数

1. cmpc

功能：输入输出受限的模型预测控制器设计与仿真。

语法：yp = cmpc(plant,model,ywt,uwt,M,P,tend,r)

[yp,u,ym] = cmpc(plant,model,ywt,uwt,M,P,tend, r,ulim,ylim,tfilter,dplant,dmodel,dstep)

说明：输入参数定义：

plant：对象的实际 MPC 阶跃响应模型；

model：辨识的 MPC 阶跃响应模型；

ywt：二次型性能指标的输出误差加权矩阵，如果 ywt 非空，则 ywt 必须有 ny 列，其中 ny 为系统输出的个数；

uwt：二次型性能指标的控制量加权矩阵，如果 uwt 非空，则 uwt 必须有 nu 列，其中 nu 为系统输入的个数；

M：控制时域长度；

P：预测时域长度；

tend：仿真的结束时间；

r：输出设定值或参考轨迹，

$$\mathrm{r}=\begin{bmatrix} r_1(1) & r_2(1) & \cdots & r_{n_y}(1) \\ r_1(2) & r_2(2) & \cdots & r_{n_y}(2) \\ \vdots & \vdots & \cdots & \vdots \\ r_1(N) & r_2(N) & \cdots & r_{n_y}(N) \end{bmatrix}$$

其中 $r_i(k)$ 为 $t=kT$ 时刻的第 i 个输出，T 为采样周期。

ulim：输入控制变量的约束矩阵，包括输出变量的上下界轨迹，其形式如：

$$\mathrm{ulim}=\left[\begin{bmatrix} u_{\min,1}(1) & \cdots & u_{\min,n_v}(1) \\ u_{\min,1}(2) & \cdots & u_{\min,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\min,1}(N) & \cdots & u_{\min,n_v}(N) \end{bmatrix}\begin{bmatrix} u_{\max,1}(1) & \cdots & u_{\max,n_v}(1) \\ u_{\max,1}(2) & \cdots & u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\max,1}(N) & \cdots & u_{\max,n_v}(N) \end{bmatrix}\begin{bmatrix} \Delta u_{\max,1}(1) & \cdots & \Delta u_{\max,n_v}(1) \\ \Delta u_{\max,1}(2) & \cdots & \Delta u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ \Delta u_{\max,1}(N) & \cdots & \Delta u_{\max,n_v}(N) \end{bmatrix}\right]$$

第一个矩阵给出下界，第二个矩阵给出上界，第三个矩阵给出扰动变量的扰动极限；

ylim：输出变量的约束矩阵，包括上下界的轨迹，缺省为正负无穷；

tfilter：噪声滤波器的时间常数，和未测扰动的滞后时间常数，缺省值对应无滤波器和类似阶跃的未测扰动情形；

dplant：输入扰动（所有可测或不可测）模型的阶跃响应系数矩阵；

dmodel：输入可测扰动模型的阶跃响应系数矩阵；

dstep：对系统的扰动，当指定 dplant 时，必须指定 dstep。

输出参数定义：

y：系统的输出；

u：控制变量；

ym：模型预测输出。

【例 1】 考虑如下的线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
ny=2;
tfinal=90;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
plant=model;
%预测时域长度为 6，控制时域长度为 2
P=6;
M=2;
ywt=[ ];
uwt=[1 1];
tend=30;
r=[0 1];
ulim=[-inf -0.15 inf inf 0.1 100];
ylim=[ ];
[y,u]=cmpc(plant,model,ywt,uwt,M,P,tend,r,ulim,ylim);
plotall(y,u,delt)
```

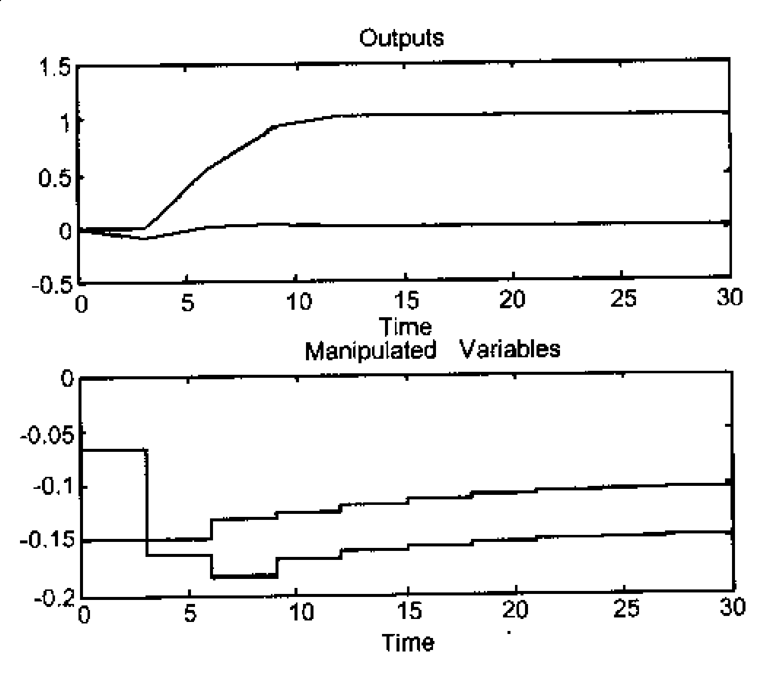


图 5-11　闭环系统输出曲线及控制量变化曲线

2. mpccl

功能：计算模型预测控制系统的闭环模型。

语法：[clmod] = mpccl(plant,model,Kmpc)

[clmod,cmod] = mpccl(plant,model,Kmpc,tfilter,dplant, dmodel)

说明：输入参数

plant：对象的实际 MPC 阶跃响应模型；

model：用来设计 MPC 控制器的 MPC 阶跃响应模型；

Kmpc：模型预测控制器的增益矩阵；

tfilter：滤波器的时间常数和噪声动力学参数构成的矩阵；

dplant：表示系统所有扰动的阶跃响应模型；

dmodel：可测扰动的阶跃响应模型。

输出参数 clmod：模型预测控制闭环系统的 MPC mod 模型；

cmod：控制器的 MPC mod 模型。

【例 2】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3; ny=2; tfinal = 60;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
plant=model;
%预测时域长度为 6
P=6;
%控制时域长度为 2
M=2;
%设置性能指标的加权矩阵
ywt=[ ];
uwt=[ ];
Kmpc=mpccon(model,ywt,uwt,M,P);
clmod=mpccl(plant,model,Kmpc);
tend=30;
clstep=mod2step(clmod, tend);
plotstep(clstep)
```

这里显示前 4 个阶跃响应曲线，如图 5-12 所示。

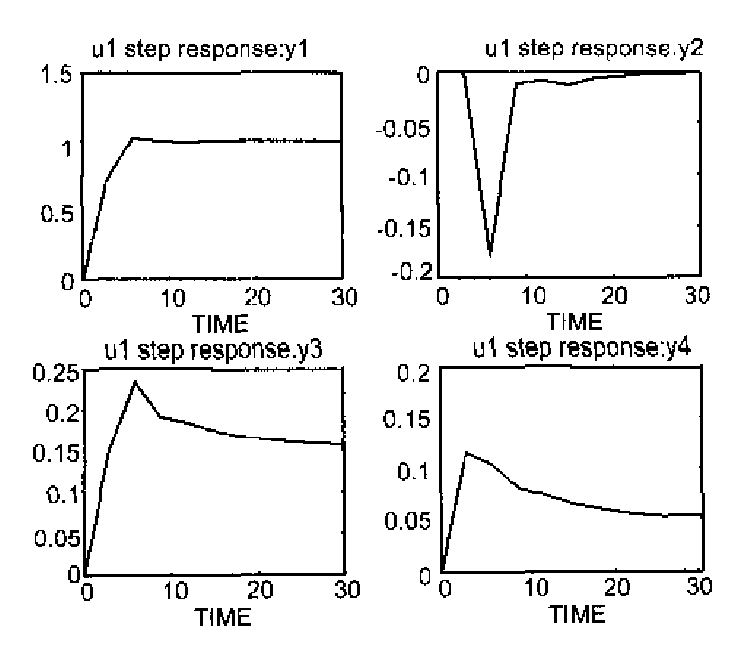


图 5-12　前 4 个阶跃响应曲线

3. mpccon

功能：输入输出不受限的模型预测控制器设计。

语法：Kmpc = mpccon(model)

　　Kmpc = mpccon(model,ywt,uwt,M,P)

说明：输入参数定义为：

model：开环对象的阶跃响应模型；

ywt：二次型性能指标的输出误差加权矩阵；

uwt：二次型性能指标的控制量加权矩阵；

M：控制时域长度；

P：预测时域长度，当 P=inf 时，表示无限的预测和控制时域长度。

输出参数定义为：

Kmpc：模型预测控制器的增益矩阵。

【例 3】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8\mathrm{e}^{-s}}{16.7s+1} & \dfrac{-18.9\mathrm{e}^{-3s}}{21.0s+1} \\ \dfrac{6.6\mathrm{e}^{-7s}}{10.9s+1} & \dfrac{-19.4\mathrm{e}^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

该系统的阶跃响应模型进行输入输出无约束的模型预测控制器设计由下面的 MATLAB 程序给出：

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
```

```
ny=2;
tfinal = 60;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
plant=model;
P=6;
M=[2 4];
uwt=[1 0; 0 1];
ywt=[1 0.1; 0.8 0.1; 0.1 0.1];
Kmpc=mpccon(model,ywt,uwt,M,P);
tend=30;
r=[1 0];
[y,u]=mpcsim(plant,model,Kmpc,tend,r);
plotall(y,u,delt)
```

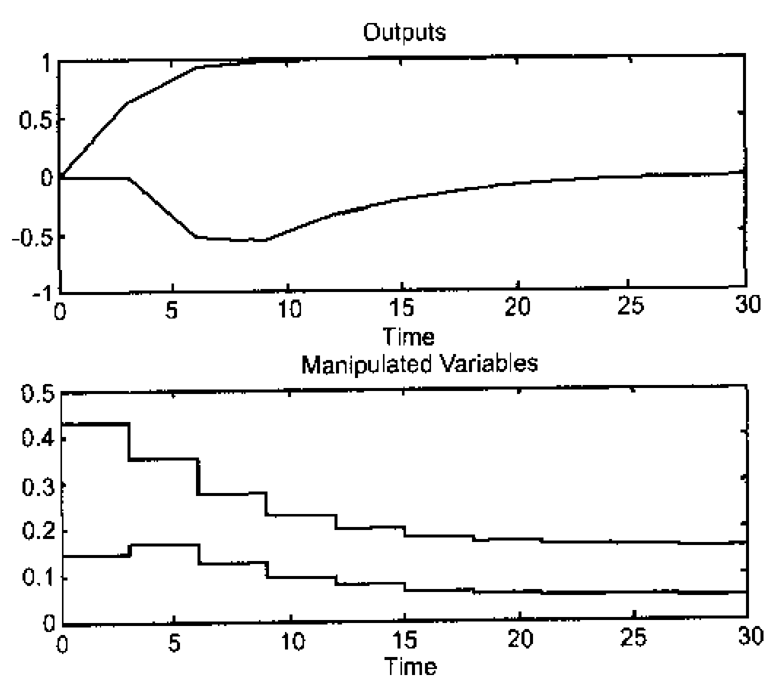


图 5-13　闭环系统输出曲线和控制量变化曲线

4. mpcsim

功能：输入输出不受限的模型预测闭环控制系统仿真。

语法：yp = mpcsim(plant,model,Kmpc,tend,r)

[yp,u,ym] = mpcsim(plant,model,Kmpc,tend,r,usat, tfilter,dplant,dmodel,dstep)

说明：输入参数定义为：

plant：开环对象的实际阶跃响应模型；

model：辨识得到的开环对象阶跃响应模型；

Kmpc：模型预测控制器的增益矩阵；

Tend：仿真的结束时间；

r：输出设定值或参考轨迹；

$$\mathrm{r}=\begin{bmatrix} r_1(1) & r_2(1) & \cdots & r_{n_y}(1) \\ r_1(2) & r_2(2) & \cdots & r_{n_y}(2) \\ \vdots & \vdots & \cdots & \vdots \\ r_1(N) & r_2(N) & \cdots & r_{n_y}(N) \end{bmatrix}$$

其中 $r_i(k)$为 $t=kT$ 时刻的第 i 个输出，T 为采样周期。

（以下的输入参数为可选参数）

usat：输入控制变量的约束矩阵，包括输出变量的上下界轨迹，其形式如：

$$\mathrm{usat}=\begin{bmatrix}\begin{bmatrix} u_{\min,1}(1) & \cdots & u_{\min,n_u}(1) \\ u_{\min,1}(2) & \cdots & u_{\min,n_u}(2) \\ \vdots & \cdots & \vdots \\ u_{\min,1}(N) & \cdots & u_{\min,n_u}(N) \end{bmatrix} \begin{bmatrix} u_{\max,1}(1) & \cdots & u_{\max,n_u}(1) \\ u_{\max,1}(2) & \cdots & u_{\max,n_u}(2) \\ \vdots & \cdots & \vdots \\ u_{\max,1}(N) & \cdots & u_{\max,n_u}(N) \end{bmatrix} \begin{bmatrix} \Delta u_{\max,1}(1) & \cdots & \Delta u_{\max,n_u}(1) \\ \Delta u_{\max,1}(2) & \cdots & \Delta u_{\max,n_u}(2) \\ \vdots & \cdots & \vdots \\ \Delta u_{\max,1}(N) & \cdots & \Delta u_{\max,n_u}(N) \end{bmatrix}\end{bmatrix}$$

第一个矩阵给出下界，第二个矩阵给出上界，第三个矩阵给出扰动变量的扰动极限；

tfilter：噪声滤波器的时间常数，和未测扰动的滞后时间常数，缺省值对应无滤波器和类似阶跃的未测扰动情形；

dplant：输入扰动（所有可测或不可测）模型的阶跃响应系数矩阵；

dmodel：输入可测扰动模型的阶跃响应系数矩阵；

dstep：对系统的扰动，当指定 dplant 时，必须指定 dstep。

输出参数定义：

y：系统的输出；

u：控制变量；

ym：模型预测输出。

【例 4】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix}=\begin{bmatrix} \dfrac{12.8\mathrm{e}^{-s}}{16.7s+1} & \dfrac{-18.9\mathrm{e}^{-3s}}{21.0s+1} \\ \dfrac{6.6\mathrm{e}^{-7s}}{10.9s+1} & \dfrac{-19.4\mathrm{e}^{-3s}}{14.4s+1} \end{bmatrix}\begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

下面的 MATLAB 程序生成模型并计算 MPC 控制器增益：

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
ny=2;
tfinal=90;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
plant=model;
```

```
P=6;
M=2;
ywt=[ ];
uwt=[1 1];
Kmpc=mpccon(model,ywt,uwt,M,P);
tend=30;
r=[0 1];
[y,u]=mpcsim(plant,model,Kmpc,tend,r);
plotall(y,u,delt)
```

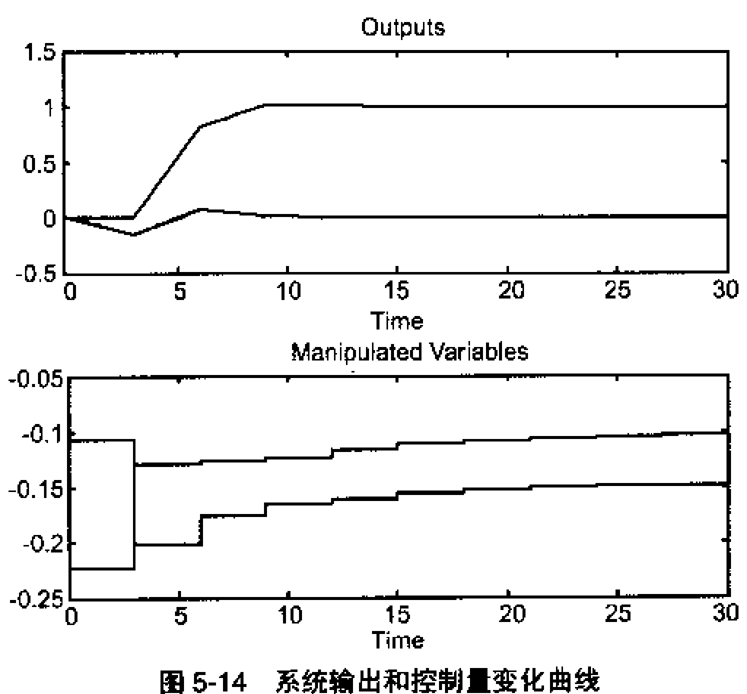


图 5-14　系统输出和控制量变化曲线

5.5.2　基于 MPC 状态空间模型的控制器设计与仿真

MALTAB 模型预测控制工具箱中除了基于模型阶跃响应的控制器设计，还提供了基于 MPC 状态空间模型（mod 模型）的控制器设计，见表 5-7。

表 5-7　基于 MPC 状态空间模型的控制器设计与仿真函数

函 数 名	函 数 功 能 说 明
scmpc	输入输出由约束的状态空间模型预测控制器设计
smpccl	计算输入输出无约束的模型预测闭环控制系统模型
smpccon	输入输出无约束的状态空间模型预测控制器设计
smpcest	状态估计器设计
smpcsim	模型预测闭环控制系统仿真

1. scmpc

功能：输入输出由约束的状态空间模型预测控制器设计。

语法：yp = scmpc(pmod,imod,ywt,uwt,M,P,tend,r)

[yp,u,ym] = scmpc(pmod,imod,ywt,uwt,M,P,tend, r,ulim,ylim,Kest,z,d,w,wu)

说明：pmod：MPC mod 格式的对象状态空间模型，用于仿真；

imod：MPC mod 格式的对象内部模型，用于预测控制器设计；

ywt：二次型性能指标的输出误差加权矩阵；

uwt：二次型性能指标的控制量加权矩阵；

M：控制时域长度；

P：预测时域长度；

tend：系统仿真所需的时域长度；

r：输出设定值或参考轨迹；

$$\mathrm{r}=\begin{bmatrix} r_1(1) & r_2(1) & \cdots & r_{n_y}(1) \\ r_1(2) & r_2(2) & \cdots & r_{n_y}(2) \\ \vdots & \vdots & \cdots & \vdots \\ r_1(N) & r_2(N) & \cdots & r_{n_y}(N) \end{bmatrix}$$

其中 $r_i(k)$为 $t=kT$ 时刻的第 i 个输出，T 为采样周期。

ulim：输入控制变量的约束矩阵，包括输出变量的上下界轨迹，其形式如：

$$\mathrm{ulim}=\left[\begin{bmatrix} u_{\min,1}(1) & \cdots & u_{\min,n_v}(1) \\ u_{\min,1}(2) & \cdots & u_{\min,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\min,1}(N) & \cdots & u_{\min,n_v}(N) \end{bmatrix}\begin{bmatrix} u_{\max,1}(1) & \cdots & u_{\max,n_v}(1) \\ u_{\max,1}(2) & \cdots & u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\max,1}(N) & \cdots & u_{\max,n_v}(N) \end{bmatrix}\begin{bmatrix} \Delta u_{\max,1}(1) & \cdots & \Delta u_{\max,n_v}(1) \\ \Delta u_{\max,1}(2) & \cdots & \Delta u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ \Delta u_{\max,1}(N) & \cdots & \Delta u_{\max,n_v}(N) \end{bmatrix}\right]$$

第一个矩阵给出下界，第二个矩阵给出上界，第三个矩阵给出扰动变量的扰动极限；

ylim：输出变量的约束矩阵，包括上下界的轨迹，缺省为正负无穷；

Kest：估计器的增益矩阵；

z：测量噪声；

d：测量扰动矩阵；

w：输出未测量扰动；

wu：施加到控制输入的未测量扰动。

输出参数定义：

y：系统的输出；

u：控制变量；

ym：模型预测输出。

【例 1】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix}=\begin{bmatrix} \dfrac{12.8\mathrm{e}^{-s}}{16.7s+1} & \dfrac{-18.9\mathrm{e}^{-3s}}{21.0s+1} \\ \dfrac{6.6\mathrm{e}^{-7s}}{10.9s+1} & \dfrac{-19.4\mathrm{e}^{-3s}}{14.4s+1} \end{bmatrix}\begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

其 MATLAB 程序：

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
pmod=imod;
P=6;
M=2;
ywt=[ ];
uwt=[1 1];
tend=30;
r=[0 1];
ulim=[-inf -0.15 inf inf 0.1 100];
ylim=[ ];
[y,u]=scmpc(pmod,imod,ywt,uwt,M,P,tend,r,ulim,ylim);
plotall(y,u,delt)
```

闭环系统输出和控制量曲线如图 5-15 所示。

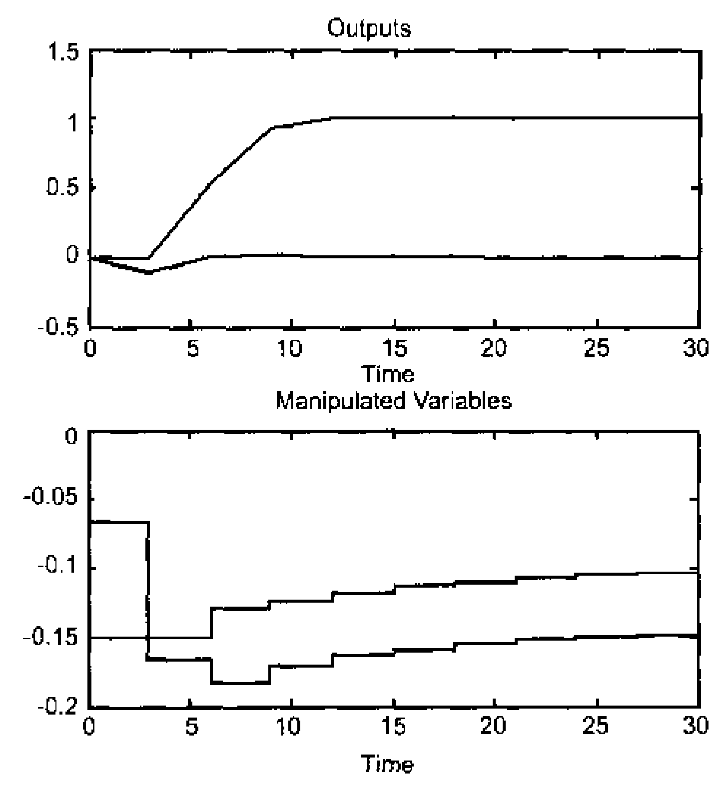


图 5-15　闭环系统输出和控制量曲线

在添加对输出变量的约束后，进行模型预测控制器的设计，其 MATLAB 程序为：

```
ulim=[-inf -0.15 inf inf 0.1 100];
ylim=[0 0 inf inf];
[y,u]=scmpc(pmod,imod,ywt,uwt,M,P,tend,r,ulim,ylim);
plotall(y,u,delt)
```

输出控制量变化曲线如图 5-16 所示。

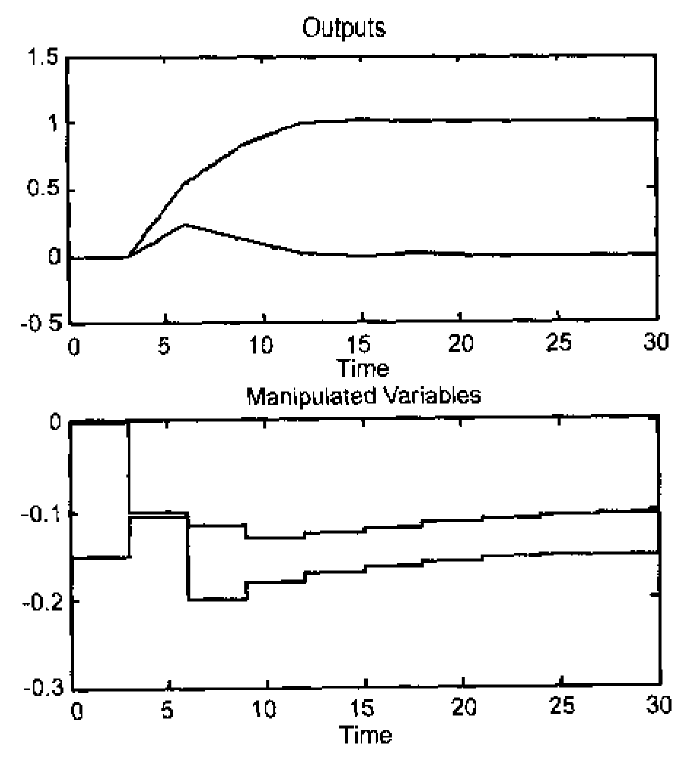


图 5-16 输出有约束时的输出响应和控制量变化曲线

2. smpccl

功能：计算输入输出无约束的模型预测闭环控制系统模型。

语法：[clmod,cmod] = smpccl(pmod,imod,Ks)

[clmod,cmod] = smpccl(pmod,imod,Ks,Kest)

说明：pmod：MPC mod 格式的对象状态空间模型；

imod：MPC mod 格式的对象内部模型；

Ks：预测控制器的增益矩阵；

Kest：状态估计器的增益矩阵。

输出参数定义为：

clmod：闭环系统的状态空间模型（MRC mod 格式）；

cmod：预测控制器的状态空间模型（MRC mod 格式）。

【例 2】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

其 MATLAB 程序：

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
pmod=imod;
P=6;
M=2;
ywt=[ ];
uwt=[ ];
Ks=smpccon(imod,ywt,uwt,M,P);
clmod=smpccl(pmod,imod,Ks);
maxpole=max(abs(smpcpole(clmod)))
tend=30;
clstep=mod2step(clmod,tend);
plotstep(clstep)
```

输入输出无约束的系统阶跃响应曲线如图 5-17 所示。

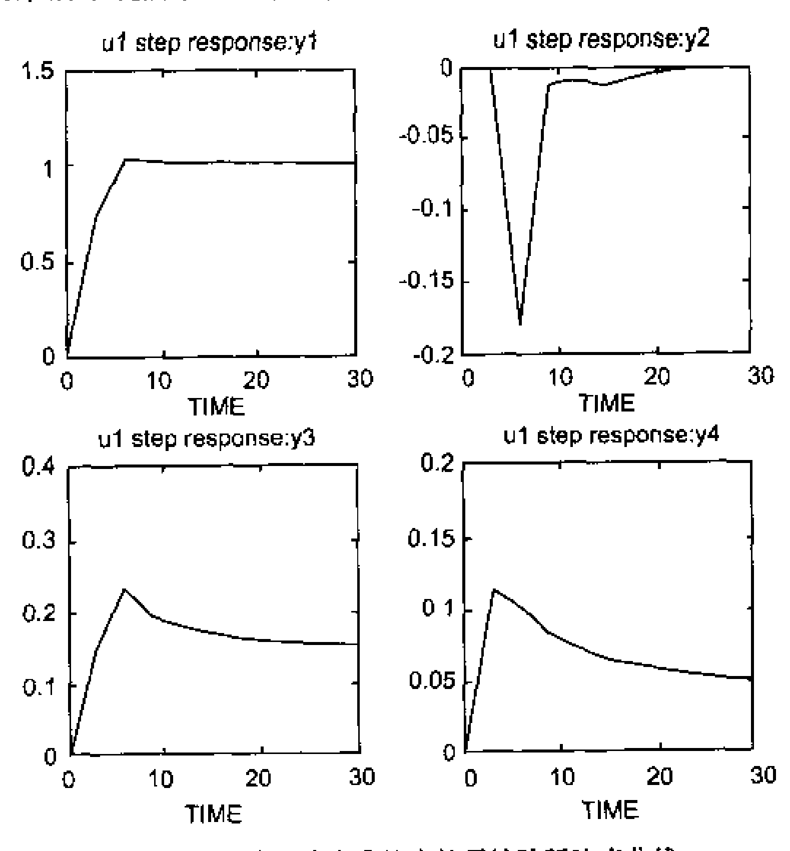


图 5-17　输入输出无约束的系统阶跃响应曲线

3. smpccon

功能：输入输出无约束的状态空间模型预测控制器设计。

语法：Ks = smpccon(imod)

Ks = smpccon(imod,ywt,uwt,M,P)

说明：输入参数定义：imod 为用于控制器设计的 MPC mod 模型；

ywt：二次型性能指标的输出误差加权矩阵；

uwt：二次型性能指标的控制量加权矩阵；

M：控制时域长度；

P：预测时域长度；

输出参数 Ks 为预测控制器的增益矩阵。

【例 3】 考虑线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8\mathrm{e}^{-s}}{16.7s+1} & \dfrac{-18.9\mathrm{e}^{-3s}}{21.0s+1} \\ \dfrac{6.6\mathrm{e}^{-7s}}{10.9s+1} & \dfrac{-19.4\mathrm{e}^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

则生成控制器的 MATLAB 程序为：

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3; ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
pmod=imod;
P=6; M=[2 4];
uwt=[1 0; 0 1];
ywt=[1 0.1; 0.8 0.1; 0.1 0.1];
Ks=smpccon(imod,ywt,uwt,M,P);
tend=30; r=[1 0];
[y,u]=smpcsim(pmod,imod,Ks,tend,r);
plotall(y,u,delt)
```

系统输出响应曲线和控制量变化曲线如图 5-18 所示。

4. smpcest

功能：状态估计器设计。

语法：对于一般的情形：

[Kest] = smpcest(imod,Q,R)

对于简化的扰动情形：

[Kest,newmod] = smpcest(imod)

[Kest,newmod] = smpcest(imod,tau,signoise)

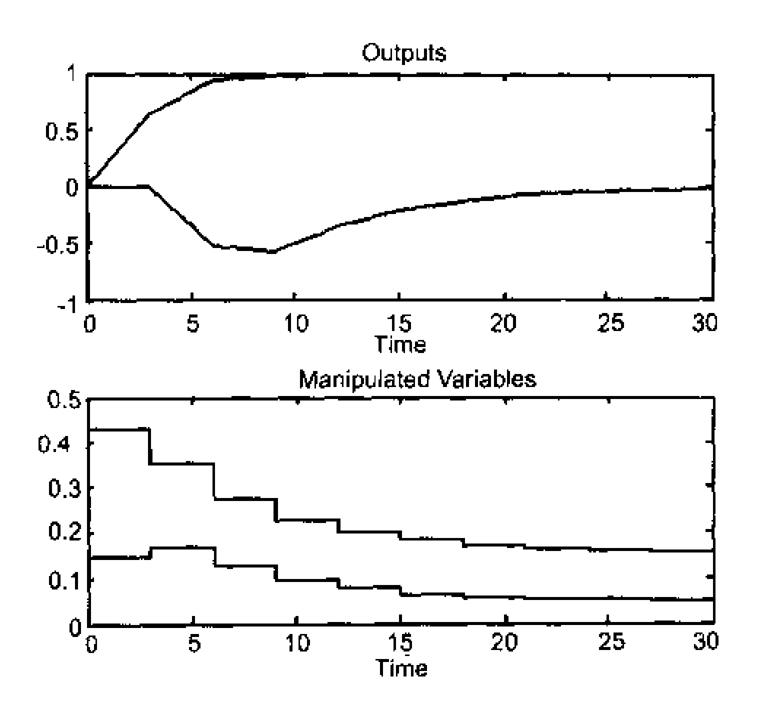


图 5-18　输入输出无约束时系统输出响应和控制量变化曲线

说明：输入参数定义：

imod：系统 MPC mod 模型；

Q：未测量扰动方差矩阵；

R：测量噪声方差矩阵；

tau：长度为 ny 的行向量，各个分量用于指定扰动对每个输出影响的特性，每个值为非负常数，可以为 inf，缺省为 0 向量。ny 为系统输出个数；

signoise：长度为 ny 的行向量，给出每个扰动输出的信噪比，缺省为 inf。

输出参数：

Kest：状态估计器增益矩阵；

newmod：修改了 imod 模型后的系统模型，在该模型中使用新的状态来表示扰动的影响。

【例 4】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \begin{bmatrix} \dfrac{3.8e^{-8s}}{14.9s+1} \\ \dfrac{4.9e^{-3s}}{13.2s+1} \end{bmatrix} w(s)$$

以下语句生成两个系统 pmod 和 imod，pmod 中包含扰动的模型，而 imod 中不包含。

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=1; ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
gw1=poly2tfd(3.8,[14.9 1],0,8);
```

```
gw2=poly2tfd(4.9,[13.2 1],0,3);
pmod=addumd(imod,tfd2mod(delt,ny,gw1,gw2));
%设计模型预测控制器
P=6; M=2;
ywt=[ ]; uwt=[1 1];
Ks=smpccon(imod,ywt,uwt,M,P);
%设计模型状态估计器
Kest1=smpcest(pmod,1,0.001*eye(ny));
Ks1=smpccon(pmod,ywt,uwt,M,P);
%生成另一个简化的估计器
r=[ ]; ulim=[ ]; z=[ ]; d=[ ]; w=[1]; wu=[ ]; tend=30;
tau=[10 10]; signoise=[3 3];
[Kest2,newmod]=smpcest(imod,tau,signoise);
Ks2=smpccon(newmod,ywt,uwt,M,P);
[y,u]=smpcsim(pmod,newmod,Ks2,tend,r,ulim,Kest2, z,d,w,wu);
plotall(y,u,2)
```

执行程序，系统输出如图 5-19 所示。

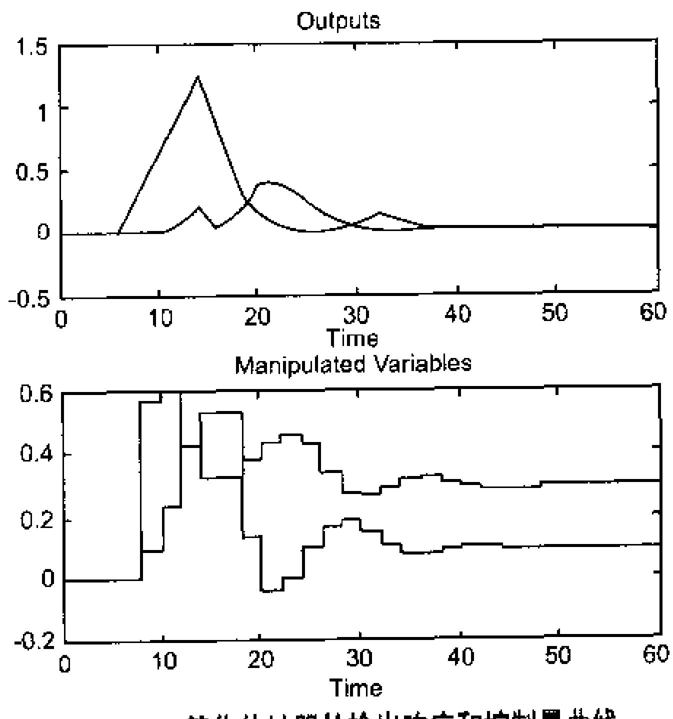


图 5-19　简化估计器的输出响应和控制量曲线

5. smpcsim

功能：模型预测闭环控制系统仿真。

语法：yp = smpcsim(pmod,imod,Ks,tend,r)

[yp,u,ym] = smpcsim(pmod,imod,Ks,tend,r,usat,Kest,z,d,w,wu)

说明：输入参数定义：

pmod：MPC mod 格式的对象状态空间模型，用仿真；



imod：MPC mod 格式的内部模型，用预测控制器设计；

Ks：预测控制器的增益矩阵；

tend：仿真时间长度；

r：输出设定值或参考轨迹；

$$r=\begin{bmatrix} r_1(1) & r_2(1) & \cdots & r_{n_y}(1) \\ r_1(2) & r_2(2) & \cdots & r_{n_y}(2) \\ \vdots & \vdots & \cdots & \vdots \\ r_1(N) & r_2(N) & \cdots & r_{n_y}(N) \end{bmatrix}$$

其中 $r_i(k)$为 $t=kT$ 时刻的第 i 个输出，T 为采样周期。

ulim：输入控制变量的约束矩阵，包括输出变量的上下界轨迹，其形式如下：

$$\text{ulim}=\left[\begin{bmatrix} u_{\min,1}(1) & \cdots & u_{\min,n_v}(1) \\ u_{\min,1}(2) & \cdots & u_{\min,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\min,1}(N) & \cdots & u_{\min,n_v}(N) \end{bmatrix} \begin{bmatrix} u_{\max,1}(1) & \cdots & u_{\max,n_v}(1) \\ u_{\max,1}(2) & \cdots & u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ u_{\max,1}(N) & \cdots & u_{\max,n_v}(N) \end{bmatrix} \begin{bmatrix} \Delta u_{\max,1}(1) & \cdots & \Delta u_{\max,n_v}(1) \\ \Delta u_{\max,1}(2) & \cdots & \Delta u_{\max,n_v}(2) \\ \vdots & \cdots & \vdots \\ \Delta u_{\max,1}(N) & \cdots & \Delta u_{\max,n_v}(N) \end{bmatrix}\right]$$

第一个矩阵给出下界，第二个矩阵给出上界，第三个矩阵给出扰动变量的扰动极限，ulim=[Ulow Uhigh delU]；

Kest：估计器增益矩阵；

z：测量噪声；

d：可测扰动；

w：输出不可测扰动；

wu：输入不可测扰动；

输出参数：

yp：系统响应矩阵，包含 M 行 ny 列，其中 $M = max(fix(tend=T) + 1, 2)$；

u：控制变量矩阵，包含与 yp 相同的列，包含 nu 行；

ym：模型预测输出，和 yp 具有相同的结构。

【例 5】 考虑如下的线性模型(图 5-20)：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix}=\begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix}\begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3; ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
pmod=imod;
P=6; M=2;
```

```
ywt=[ ]; uwt=[1 1];
Ks=smpccon(imod,ywt,uwt,M,P);
tend=30; r=[0 1];
[y,u]=smpcsim(pmod,imod,Ks,tend,r);
plotall(y,u,delt),pause
```

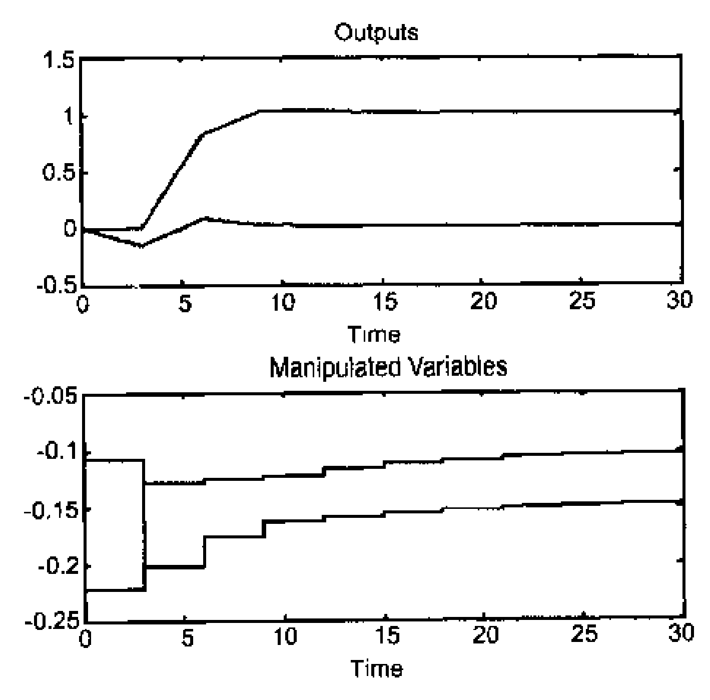


图 5-20 闭环系统输出和控制量变化曲线

%修改输入 $u1$ 扰动的脉冲，见图 5-21。

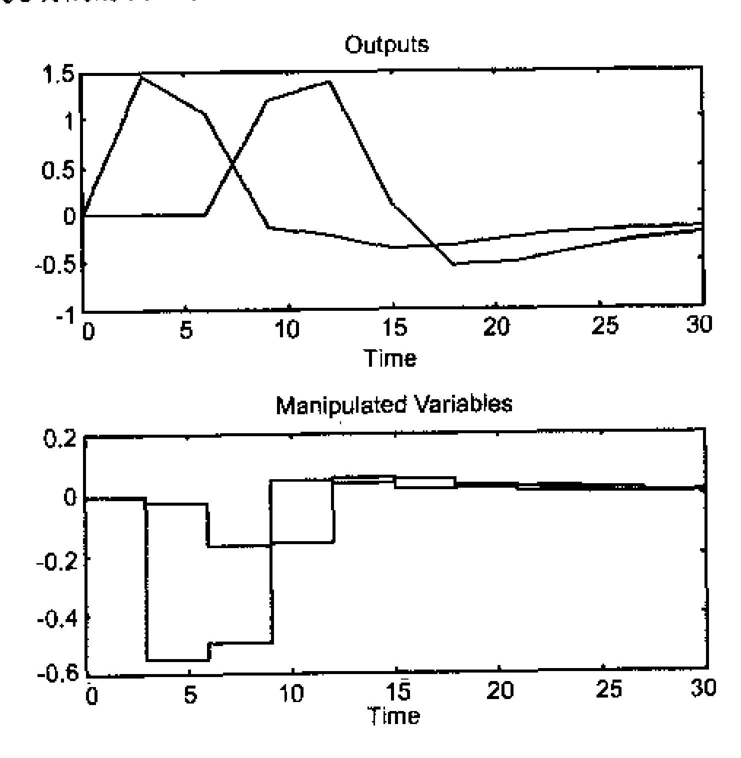


图 5-21 改变扰动后的系统响应及控制量变化曲线

```
r=[ ]; usat=[ ]; Kest=[ ]; z=[ ]; d=[ ]; w=[ ];
wu=[ 1 0; 0 0];
[y,u]=smpcsim(pmod,imod,Ks,tend,r,usat,Kest,z,d,w,wu);
plotall(y,u,delt),pause
%修改两个输入扰动的脉冲，如图 5-22 所示。
usat=[-inf -inf inf inf 0.1 0.05];
[y,u]=smpcsim(pmod,imod,Ks,tend,r,usat,Kest,z,d,w,wu);
plotall(y,u,delt),pause
```

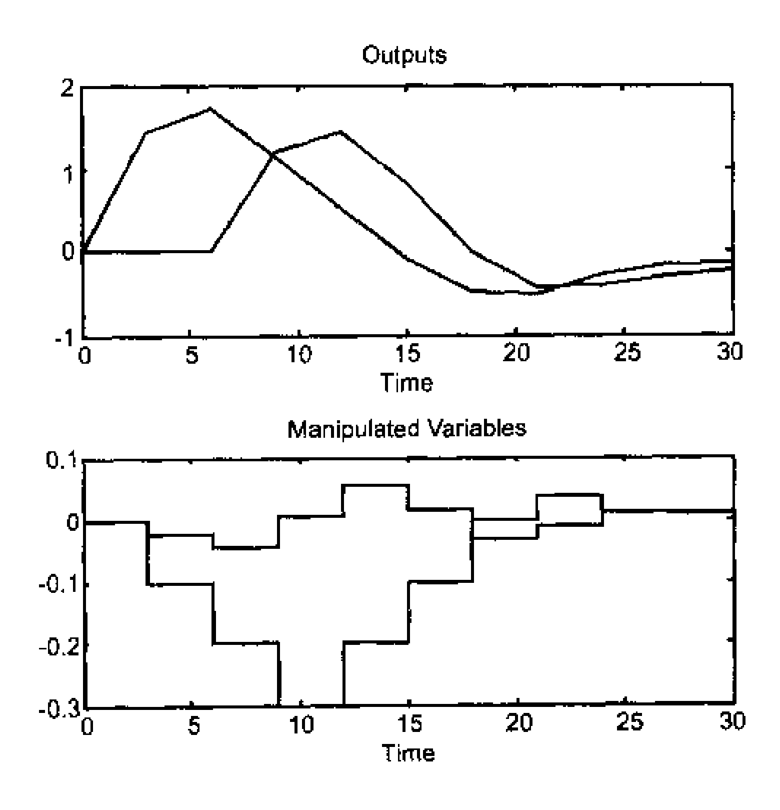


图 5-22　改变两个输入的扰动后的系统响应及控制量变化曲线

5.6　系统分析函数

MALTAB 模型预测工具箱提供了对模型特性分析的函数，见表 5-8。

表 5-8　系统分析函数列表说明

函 数 名	函 数 功 能 说 明
mod2frsp	计算系统（MPC mod）的频率响应
smpcgain	计算系统（MPC mod）的稳态增益矩阵
smpcpole	计算系统（MPC mod）的极点
svdfrsp	计算频率响应的奇异值

1．mod2frsp

功能：计算系统（MPC mod）的频率响应。

语法：frsp = mod2frsp(mod,freq)

[frsp,eyefrsp] = mod2frsp(mod,freq,out,in,balflg)

说明：输出参数定义：

mod：系统 MPC mod 模型；

freq：频率向量，指定对频率区间范围内的频率点的个数；

out：指定输出变量，如果 out=[]，则计算所有输出；

in：指定输入变量，如果 in=[]，则计算所有输入；

balflg：在计算前进行 PHI 矩阵的均衡处理；

输出参数：

frsp：系统的输出频率响应矩阵；

eyefrsp：当频率响应矩阵为方阵时，eyefrsp=I-frsp，I 为单位矩阵。

【例 1】 考虑如下线性系统：

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3; ny=2; tfinal = 60;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
plant=model;
P=6;
M=2;
ywt=[ ];
uwt=[ ];
Kmpc=mpccon(model,ywt,uwt,M,P);
clmod=mpccl(plant,model,Kmpc);
freq = [-3,0,30];
in = [1:ny];
out = [1:ny];
[frsp,eyefrsp] = mod2frsp(clmod,freq,out,in);
plotfrsp(eyefrsp);
pause;
%以上代码生成图 5-23。
plotfrsp(frsp);
```






```
pause;
%以上代码生成图 5-24。
```

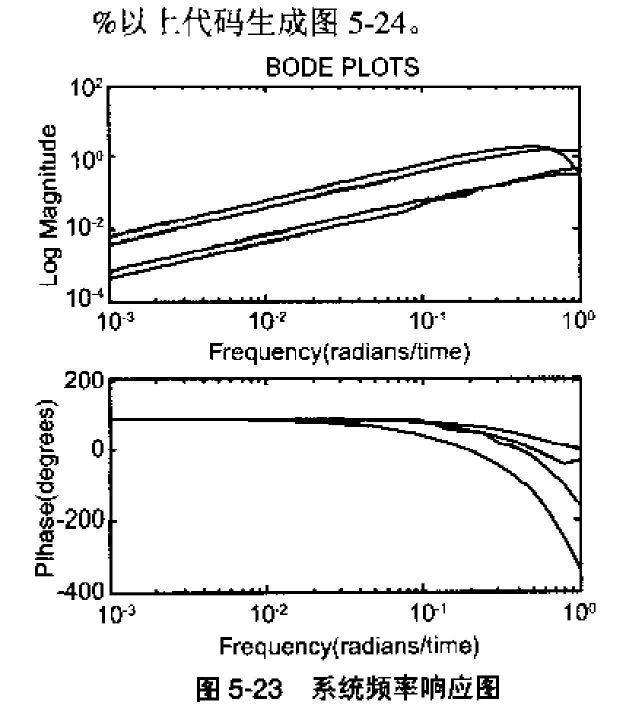


图 5-23　系统频率响应图

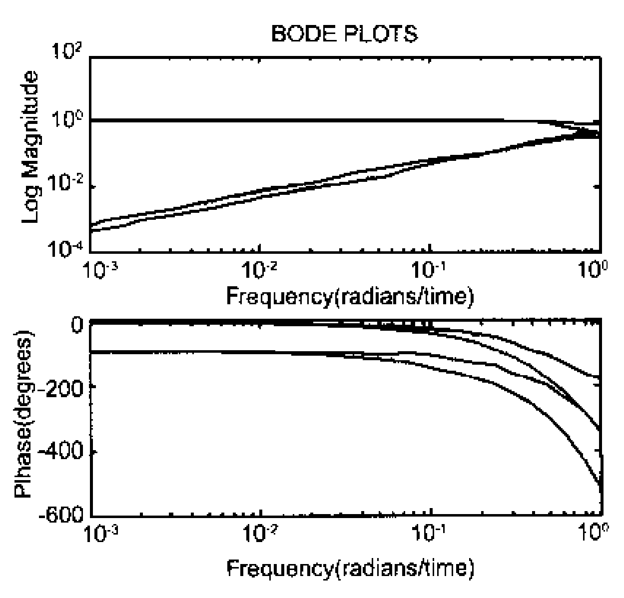


图 5-24　系统频率响应波特图

```
[sigma,omega] = svdfrsp(eyefrsp);
clg;
semilogx(omega,sigma);
title('Singular Values vs. Frequency');
xlabel('Frequency (radians/time)');
ylabel('Singular Values');
%以上代码生成图 5-25。
```

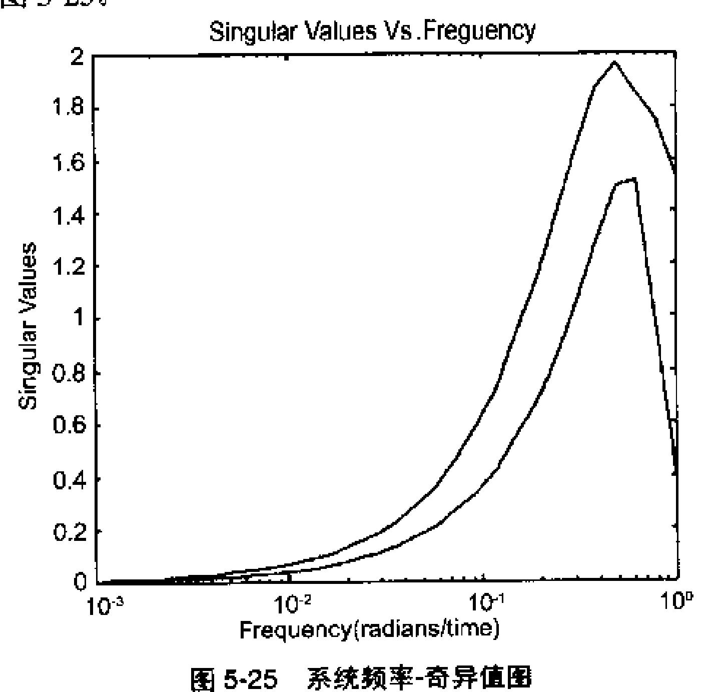


图 5-25　系统频率-奇异值图

2. smpcgain

功能：计算（MPC mod）系统的稳态增益矩阵。

语法：g = smpcgain(mod)

说明：mod 为 MPC mod 格式的动态模型。

输出参数 g 为系统的稳态增益矩阵，行数为输出变量数，列数等于输入变量数。

3. smpcpole

功能：计算系统（MPC mod）的极点。

语法：poles = smpcpole(mod)

说明：mod 为系统的 MPC mod 模型；

输出参数 poles 为复数向量给出的系统极点。

4. svdfrsp

功能：计算频率响应的奇异值。

语法：[sigma,omega] = svdfrsp(vmat)

说明：输入参数 vmat：系统的频率响应矩阵；

输出参数 sigma：频率响应的奇异值矩阵，包含矩阵函数 $F(\omega)$的采样值：

$F(\omega_1), F(\omega_2),\ldots, F(\omega_N)$，如果 $F(\omega_i)$的最小值为 m，且 $\sigma_1(\omega_i),\ldots,\sigma_m(\omega_i)$为 $F(\omega_i)$的降序排列的奇异值，则有：

$$\text{sigma}=\begin{bmatrix}\sigma_1(\omega_1) & \sigma_2(\omega_1) & \cdots & \sigma_m(\omega_1)\\ \sigma_1(\omega_2) & \sigma_2(\omega_2) & \cdots & \sigma_m(\omega_2)\\ \vdots & \vdots & \cdots & \vdots\\ \sigma_1(\omega_N) & \sigma_2(\omega_N) & \cdots & \sigma_m(\omega_N)\end{bmatrix}$$

omega：包含频率（$\omega_1,\ldots,\omega_N$）的独立变量值的列向量。

5.7 模型预测控制工具箱功能函数

MATLAB 模型预测工具箱还提供了许多功能函数，包括模型转换和检验函数、仿真函数和方程求解函数等，见表 5-9。

表 5-9 模型预测控制工具箱功能函数列表说明

函 数 名	函 数 功 能 说 明
abcdchkm	检查状态空间矩阵(A,B,C,D)维数的一致性（等价于控制系统工具箱的 abcdchk）
dantzgmp	求解二次规划问题
dareiter	求解离散 Riccati 方程
dimpulsm	生成离散系统的脉冲响应（等价于控制系统工具箱的 dimpulse）
dlqe2	计算离散系统的状态估计器增益矩阵
dlsimm	离散系统仿真（等价于控制系统工具箱的 dlsim）
mpcaugss	增广状态空间模型
mpcparal	将两个状态空间模型并联
nargchkm	检测 M 文件的变量个数（等价于控制系统工具箱的 nargchk）
mpcstair	生成接替型控制变量
vec2mat	将向量转换成矩阵

由于许多函数在前面控制系统工具箱中都有详细说明，故这里不给出详细说明。下面给出跟本章联系紧密的几个函数的说明。

1．dlqe2

功能：通过求解离散 Riccati 方程，计算系统的状态估计器增益矩阵。

语法：k = dlqe2(phi,gamw,c,q,r)

[k,m,p] = dlqe2(phi,gamw,c,q,r)

说明：该函数通过迭代法求解离散的 Kalman 滤波器方程。

输入参数定义：

phi, c：对象的 MPC 状态空间矩阵；

gamw：不可测扰动的特性矩阵；

q：不可测扰动的方差矩阵；

r：测量噪声方差矩阵。

输出参数定义：

k：离散 Kalman 滤波器的稳态增益矩阵；

m：测量更新之前的状态估计期望方差；

p：测量更新之后的状态估计期望方差；

【例 1】 考虑由图 5-26 给出的系统：

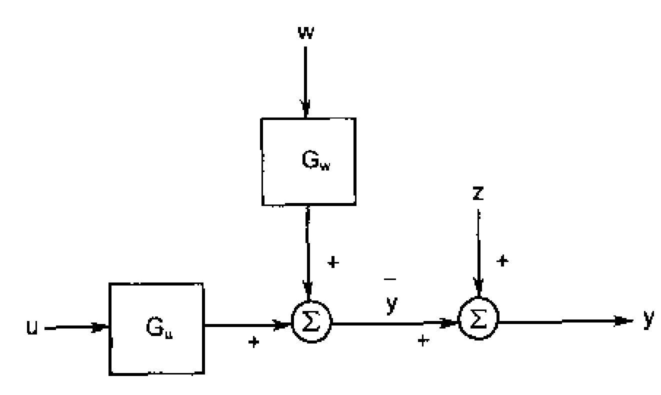


图 5-26　模型系统框图表示

其中 G_u 和 G_w 为一阶离散传递函数：

$$G_u(z)=\frac{0.20}{1-0.8z^{-1}} \qquad G_w(z)=\frac{0.3}{1-0.95z^{-1}}$$

未测量输入：Q = 2, R = 1。

下面利用 MPC 工具箱为上述系统建模，并计算系统增益：

```
delt=2; ny=1;
gu=poly2tfd(0.2,[1 -0.8],delt);
Gw=poly2tfd(0.3,[1 -0.95],delt);
[phi,gam,c,d]=mod2ss(tfd2mod(delt,ny,gu,Gw));
k=dlqe2(phi,gam(:,2),c,2,1)
```

MATLAB 返回：

```
k = 0
```

1.0619

2. mpcaugss

功能：增广状态空间模型。

语法：[phia,gama,ca] = mpcaugss(phi,gam,c)

[phia,gama,ca,da,na] = mpcaugss(phi,gam,c,d)

说明：输入参数：

phi,gam,c,d：系统 MPC 状态空间模型矩阵；

输出参数：

phia,gama,ca,da：增广系统的 MPC 状态空间模型矩阵；

na：增广系统的阶次。

【例 2】 生成一个有 2 状态,3 输入,2 输出的系统：

```
phi=diag([0.8, -0.2]);
gam=[1 -1 0;0 2 -0.5];
c=[0.4 0;0 1.5];
%增广该系统
[phia,gama,ca,da,na]=mpcaugss(phi,gam,c)
```

MATLAB 返回：

```
phia =
    0.8000    0         0         0
    0         -0.2000   0         0
    0.3200    0         1.0000    0
    0         -0.3000   0         1.0000
gama =
    1.0000    -1.0000   0
    0         2.0000    -0.5000
    0.4000    -0.4000   0
    0         3.0000    -0.7500
ca =
    0    0    1    0
    0    0    0    1
da =
    0    0    0
    0    0    0
na =
4
```

3. vec2mat

功能：将向量转换为矩阵。

语法：mat = vec2mat(vec,matcol);

mat = vec2mat(vec,matcol,padding);

```
[mat,padded] = vec2mat(...);
```

说明：函数将向量 vec 转换为具有 matcol 列的矩阵 mat，如果向量的长度不是 matcol 的整数倍，则其余补上 0。

可选参数 padding 说明当向量的长度不是 matcol 的整数倍，则其余补上的元素为 padding 中的；如果 padding 长度仍然不够，则重复填入 padding 中元素。

第6章 模糊逻辑工具箱

现代数学是建立在集合论的基础上。集合论的重要意义从一个侧面看，在于它把数学的抽象能力延伸到人类认识过程的深处。一组对象确定一组属性，人们可以通过说明属性来说明概念（内涵），也可以通过指明对象来说明它。符合概念的那些对象的全体叫做这个概念的外延，外延其实就是集合。从这个意义上讲，集合可以表现概念，而集合论中的关系和运算又可以表现判断和推理，一切现实的理论系统都可能纳入集合描述的数学框架。

但是，数学的发展也是阶段性的。经典集合论只能把自己的表现力限制在那些有明确外延的概念和事物上，它明确地限定：每个集合都必须由明确的元素构成，元素对集合的隶属关系必须是明确的，决不能模棱两可。对于那些外延不分明的概念和事物，经典集合论是暂时不去反映的，属于待发展的范畴。

在较长时间里，精确数学及随机数学在描述自然界多种事物的运动规律中，获得显著效果。但是，在客观世界中还普遍存在着大量的模糊现象。以前人们回避它，但是，由于现代科技所面对的系统日益复杂，模糊性总是伴随着复杂性出现。

各门学科，尤其是人文、社会学科及其他“软科学”的数学化、定量化趋向把模糊性的数学处理问题推向中心地位。更重要的是，随着电子计算机、控制论、系统科学的迅速发展，要使计算机能像人脑那样对复杂事物具有识别能力，就必须研究和处理模糊性。人与计算机相比，一般来说，人脑具有处理模糊信息的能力，善于判断和处理模糊现象。但计算机对模糊现象识别能力较差，为了提高计算机识别模糊现象的能力，就需要把人们常用的模糊语言设计成机器能接受的指令和程序，以便机器能像人脑那样简洁灵活的做出相应的判断，从而提高自动识别和控制模糊现象的效率。这样，就需要寻找一种描述和加工模糊信息的数学工具，这推动了数学家深入研究模糊数学。

1965 年，美国控制论专家、数学家 Zadeh 发表了论文《模糊集合》，标志着模糊数学这门学科的诞生。模糊数学的研究内容主要有以下三个方面：第一，研究模糊数学的理论，以及它和精确数学、随机数学的关系。第二，研究模糊语言学和模糊逻辑。第三，研究模糊数学的应用。

针对模糊数学的广泛应用，MathWorks 公司在其 MATLAB 中加入了 Fuzzy Logic 工具箱。该工具箱由长期从事模糊逻辑和模糊控制的专家和技术人员编制。并在 MATLAB 6.1 中推出了 2.1 版本。MATLAB 模糊逻辑工具箱提供了以下强大的功能：

- 图形化的系统设计界面。
- 支持模糊逻辑中的高级技术，如自适应神经-模糊推理系统（Adaptive Neural – Fuzzy Interence System，ANFIS）、用于模式识别的模糊聚类技术和模糊推理方法选择等。

- 集成的仿真和代码生成功能，实现了 MATLAB 模糊逻辑控制工具箱和 Simulink 的无缝连接，通过 Real-Time Workshop2.1 可以生成 ANSIC 原代码，有利于实际应用。
- 独立运行的模糊推理机。在用户完成模糊逻辑系统的设计后，可以将设计结果以 ASCII 码的形式保存。利用模糊逻辑工具箱的模糊推理机，可以实现模糊逻辑系统的独立运行或作为其他应用程序的一部分运行。

6.1　模糊逻辑理论简介

6.1.1　模糊集合

模糊逻辑从模糊集合的概念开始，模糊集合是由 19 世纪末德国数学家 G.Contor 建立的。在经典的集合论中，一个元素和一个集合的关系只能是两种：属于或不属于，但在模糊集合中，给定范围内元素对它的隶属关系不一定只有“是”或“否”两种情况，而是用介于 0 和 1 之间的实数来表示隶属程度，还存在中间过渡状态。比如“老人”是个模糊概念，70 岁的肯定属于老人，它的从属程度是 1，40 岁的人肯定不算老人，它的从属程度为 0，按照 Zadeh 给出的公式，55 岁属于“老”的程度为 0.5，即“半老”，60 岁属于“老”的程度 0.8。查德认为，指明各个元素的隶属集合，就等于指定了一个集合。当隶属于 0 和 1 之间的值时，就是模糊集合。

我们用几个基本的定义来开始讨论模糊集。令 X 是一个点(对象)的空间，用 x 表示 X 的一个普通元素。X 的一个模糊集(类)A 通过一个隶属度函数 $f_A(x)$来刻画，$f_A(x)$使 X 内的每一个点与区间[0，1]内的一个实数相对应，用 $f_A(x)$在点 x 的值来表示 x 在 A 内的隶属度。$f_A(x)$的值越是接近于 1，x 属于 A 内的程度就越大。当 A 是通常意义下的集合时，它的隶属函数仅仅能取 0 和 1 两个值，$f_A(x)=1$ 或者 $f_A(x)=0$，取决于 x 属于或不属于 A。因此，在这种情况下 $f_A(x)$成为我们所熟悉的集合 A 的特征函数。

我们从涉及模糊集的几个定义作为开始，这些定义显然是普通集合上相应定义的扩展。

- 空集：一个模糊集是空的当且仅当它的隶属函数在 X 上恒等于零。
- 相等：两个模糊集 A 和 B 相等(记作 $A=B$)当且仅当对所有的 x 属于 X 有 $f_A(x)=f_B(x)$ 成立。
- 包含：B 包含 A（或者等价地说，A 是 B 的子集，或者 A 小于等于 B）当且仅当 $f_A(x)\leq f_B(x)$。
- 并集：分别相应于隶属函数 $f_A(x)$和 $f_B(x)$的模糊集合 A 与 B 的并是模糊集 C，记作 $C=A\cup B$。集合 C 的隶属函数定义为：

$$f_C(x)=\max[f_A(x),f_B(x)],x\in X$$

或简写成 $f_C=f_A\vee f_B$

- 交集：分别相应于隶属函数 $f_A(x)$和 $f_B(x)$的模糊集合 A 与 B 的交集是模糊集 C，记作 $C=A\cap B$。集合 C 的隶属函数定义为：

$$f_C(x)=\min[f_A(x),f_B(x)],x\in X$$

或简写成 $f_C=f_A\wedge f_B$

6.1.2 模糊关系

经典定义的关系是明确的关系，而模糊关系就是经典关系的推广。

经典关系：对两个经典集合 X 和 Y，直积 X×Y 的子集 R 称为 A 到 B 的一个二元关系。X×X 的子集称为 X 上的一个二元关系，二元关系简称关系。

一般来讲，n 个集合的直积 $X_1 \times X_2 \times \cdots \times X_s$ 的子集称为 $X_1 \times X_2 \times \cdots \times X_s$ 上的 N 元关系。

模糊关系：直积空间 $X \times Y$ 上的模糊关系是 $X \times Y$ 的一个模糊子集 R，R 的隶属度函数 $R(x,y)$ 表示了 X 中元素 x 与 Y 中元素 y 具有这种关系的程度。X 到 X 的模糊关系称为 X 上的模糊关系。

类似的，可以定义 n 个模糊集合的关系：$X_1,\ldots,X_n$ 为 n 个集合，直积空间 $X_1 \times \cdots \times X_n$ 上的一个 n 元模糊关系 R 是指 $X_1 \times \cdots \times X_n$ 上的一个模糊子集，R 的隶属函数 $R(x_1,\ldots x_n)$ 描述了元素 $(x_1,\ldots x_n)$ 具有这种关系的程度。

6.1.3 模糊推理

逻辑学是研究概念、判断和推理形式特别是推理形式的一门科学。从 17 世纪德国数学家 Lebniz 开始，不少数学家和哲学家共同努力，把数学方法用于哲学的研究，出现了逻辑与数学相结合的一门新学科——数理逻辑。数理逻辑采用一套符号代替人们的自然语言进行表述，因而又称符号逻辑。数理逻辑在逻辑上只取真、假两个值，是一种二值逻辑。推理的方式，一般可以分为两种，一种是演绎推理，一种是归纳推理。以一般的普遍适用的原理为前提，推导到某个特殊情况作出结论的推理方法为演绎推理。而反过来，由特殊情形的前提，归纳出一般原理的结论的推理称为归纳推理。一个是从一般到特殊，一个是从特殊到一般。

数理逻辑主要研究演绎推理。演绎推理一般具有三段论法的形式做出第三个判断。一个著名的例子就是所谓的苏格拉底论述：

所有的人都是要死的，

苏格拉底是人，

所以苏格拉底总是要死的。

在三段论法中，根据两个前提的不同形式又有直言推理、假言推理和相选言推理之分。

归纳推理又称归纳法，可分为完全归纳法和不完全归纳法。完全归纳法在前提中列出全部推理的特殊情形，而得出一般化结论。这是很严格的，但是当特殊情形过多乃至无限时都难以应用。不完全归纳法仅举出全部特殊情形中的一个或几个而归纳出一般结论。

以模糊集为理论基础的模糊逻辑推理的基本形式是假言推理的模糊化，前提和结论中包含的概念由明确概念变为模糊概念，反应一般规律的大前提可以由模糊关系来表示。假言推理有两种形式：肯定前件和否定后件的假言推理形式。前者英文称 MP(Modus Ponens)，后者称 MT (Modus Tollens)：

肯定前件式，

大前提(一般规则)　　若 x 是 A 则 y 是 B

小前提(特殊证据)　　x 是 A

结论　　　　　　　y 是 B

否定前件式：

大前提(一般规则)　　若 x 是 A 则 y 是 B

小前提(特殊证据)　　　　　　y 不是 B

结论　　　　　　　x 不是 A

由于推理前提可以有更为复杂的形式，模糊逻辑推理基本形式也就相应演变成较复杂的扩充形式。作为传统假言推理的发展和扩充，模糊逻辑推理的基本形式，也可以有相应的肯定前件式和否定后件式，称为广义的肯定前件式 GMP(Generalized Modus Ponens)和广义的否定后件式 GMT(Generalized Modus Tollens)：

广义肯定前件式：

大前提　若 x 是 A　　则 y 是 B

小前提　　x 是 A

结论　　　　　y 是 B

广义否定后件式：

大前提　若 x 是 A　　则 y 是 B

小前提　　　　　　　y 不是 B

结论　　　　　x 不是 A

模糊推理系统可以分为三大类：纯模糊逻辑系统；具有模糊产生器和模糊消除器的模糊逻辑系统；Takagi - Sugeno 型模糊逻辑系统。

1．纯模糊逻辑系统

纯模糊逻辑系统的输入输出均为模糊集合。模糊输入信息经由模糊推理机按模糊规则库的处理，映射为模糊输出，如图 6-1 所示。因为输入输出均为模糊集合，所以纯模糊逻辑系统不能直接用于实际应用。

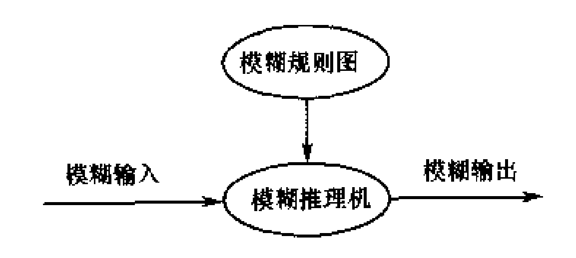


图 6-1　纯模糊逻辑系统示意图

2．具有模糊产生器和模糊消除器的模糊逻辑系统

由于在实际应用中，系统的输入和输出都是确定的值，要使用模糊逻辑系统的模糊推理机，就必须在系统输入前将输入模糊化，这就使用到了模糊产生器；再对模糊推理机的模糊输出进行去模糊处理，得到的就是精确的输出量。从而这种模糊逻辑系统可以被应用到实际工程中去。它的示意图如图 6-2 所示。这种类型的模糊逻辑系统是目前使用最为广泛的一种。

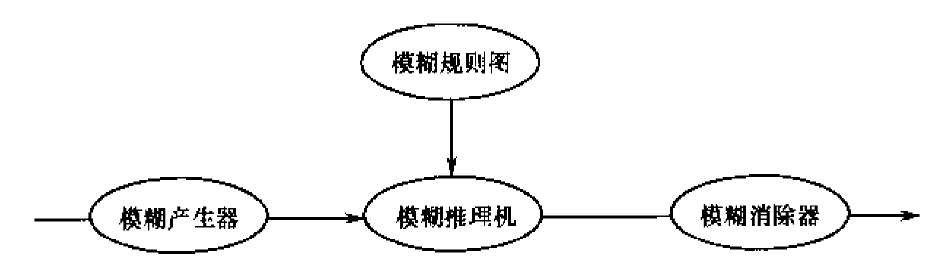


图 6-2 具有模糊产生器和模糊消除器的模糊逻辑系统示意图

3. Takagi - Sugeno 型模糊逻辑系统

Takagi - Sugeno 型模糊逻辑系统是一种特殊的模糊逻辑系统。它的输入可以为模糊集合，但输出不通过模糊消除器却可以是精确值。它的推理形式如下：

$$\text{IF} \quad x_1 \text{是} A_1,\cdots,x_n \text{是} A_n \qquad \text{THEN} \quad y=c_1x_1+\cdots+c_nx_n$$

其中 A_i 为模糊语言值，c_i 为权重。c_i 可以通过参数估计法来确定。由于输出是输入的线性关系，所以可以使用线性控制系统的分析法来分析模糊逻辑系统。但由于输出为精确的，所以不能利用专家系统控制知识，也限制了 Takagi - Sugeno 型模糊逻辑系统的应用。

6.2 MATLAB 模糊逻辑工具箱

6.2.1 模糊隶属度函数

MALTAB 模糊逻辑工具箱中提供了表 6-1 中的函数，用以生成特殊形状的隶属度函数，包括常用的高斯型、钟型、π型等隶属度函数。

表 6-1 模糊隶属度函数列表说明

函数名	说 明
dsigmf	计算两个 Sigmoid 型隶属度函数之差
gauss2mf	建立双边高斯型隶属度函数
gaussmf	建立高斯型隶属度函数
gbellmf	建立一般的钟型隶属度函数
pimf	建立π型隶属度函数
psigmf	计算两个 Sigmoid 型隶属度函数之积
smf	建立 S 型隶属度函数
sigmf	建立 Sigmoid 型隶属度函数
trapmf	建立梯形隶属度函数
trimf	建立三角形隶属度函数
zmf	建立 Z 型隶属度函数

1. dsigmf

功能：计算两个 Sigmoid 型隶属度函数之差。

语法：y = dsigmf(x,[a1 c1 a2 c2])

说明：输入参数 a1、c1、a2 和 c2 分别用于指定两个 Sigmoid 型函数的形状。

由参数 a 和 c 决定的 Sigmoid 函数为：

$$f(x;a,c)=1/(1+\exp(-a(x-c)))$$

函数返回 $f(x;a1,c1)-f(x;a2,c2)$

【例 1】 绘制两个 Sigmoid 型函数差的隶属度曲线，如图 6-3 所示。

```
x=0:0.1:10;
y=dsigmf(x,[5 2 5 7]);
plot(x,y)
xlabel('dsigmf, P=[5 2 5 7]')
```

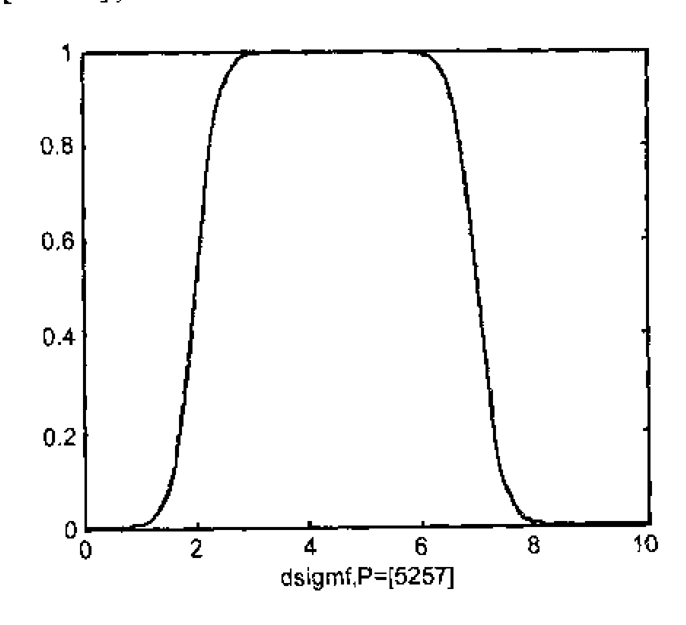


图 6-3 两个 Sigmoid 型函数之差

2. gauss2mf

功能：建立双边高斯型隶属度函数。

语法：y = gauss2mf(x,[sig1 c1 sig2 c2])

说明：高斯函数由两个参数 σ 和 c 决定：

$$f(x;\sigma,c)=\exp(-(x-c)^2/2\sigma^2)$$

由 sig1 和 c1 决定的高斯函数决定双边高斯函数的左半边曲线，由 sig2 和 c2 生成的高斯函数决定双边高斯函数的右半边曲线。当 c1<c2 时，高斯函数可以取最大值 1，否则最大值小于 1。

【例 2】 绘制多条双边高斯曲线，如图 6-4 所示。

```
x = (0:0.1:10)';
y1 = gauss2mf(x, [2 4 1 8]);
y2 = gauss2mf(x, [2 5 1 7]);
y3 = gauss2mf(x, [2 6 1 6]);
y4 = gauss2mf(x, [2 7 1 5]);
```

```
y5 = gauss2mf(x, [2 8 1 4]);
plot(x, [y1 y2 y3 y4 y5]);
set(gcf, 'name', 'gauss2mf', 'numbertitle', 'off');
```

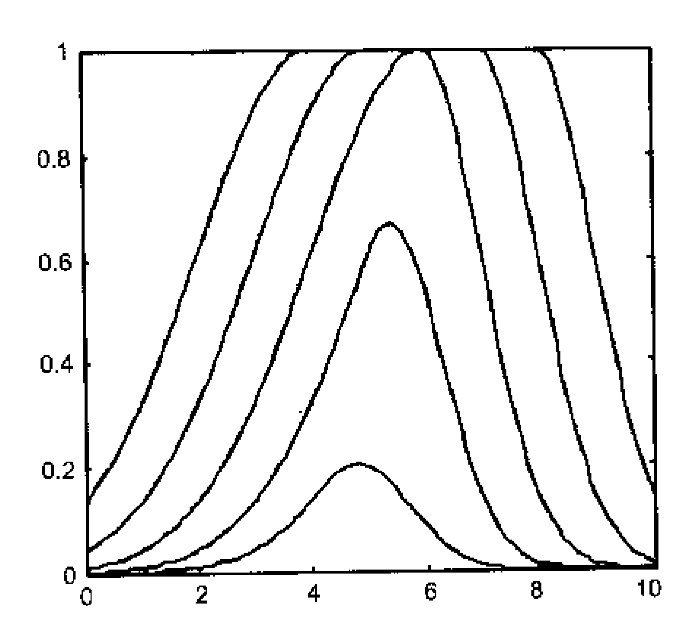


图 6-4　多条双边高斯型隶属度函数曲线

3. gaussmf

功能：建立高斯型隶属度函数。

语法：y = gaussmf(x,[sig c])

说明：函数返回以参数 sig 和 c 生成的高斯函数。

【例 3】 生成高斯型隶属度函数，如图 6-5 所示。

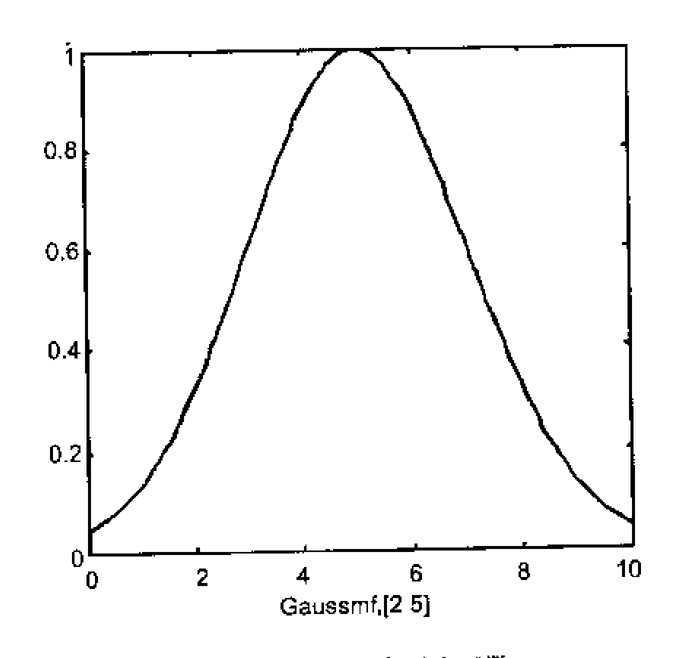


图 6-5　高斯型隶属度函数

```
x=0:0.1:10;
y=gaussmf(x,[2 5]);
plot(x,y)
xlabel('gaussmf, P=[2 5]')
```

4. gbellmf

功能：生成一般的钟型隶属度函数。

语法：y = gbellmf(x,[a b c])

说明：由参数 a,b 和 c 生成的钟型隶属度函数为

$$f(x;a,b,c)=1/(1+|(x-c)/a|^{2b})$$

【例 4】 生成一般钟型隶属度函数，如图 6-6 所示。

```
x=0:0.1:10;
y=gbellmf(x,[2 4 6]);
plot(x,y)
xlabel('gbellmf, P=[2 4 6]')
```

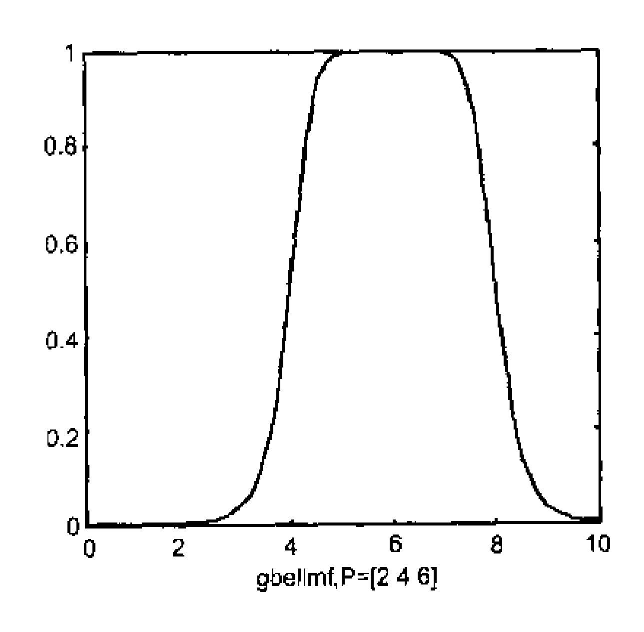


图 6-6　一般钟型隶属度函数

5. pimf

功能：建立 π 型隶属度函数。

语法：y = pimf(x,[a b c d])

说明：生成的 π 型隶属度函数为以 a,b,c 和 d 为折点的样条函数。

【例 5】 生成 π 型隶属度函数，如图 6-7 所示。

```
x=0:0.1:10;
y=pimf(x,[1 4 5 10]);
plot(x,y)
xlabel('pimf, P=[1 4 5 10]')
```

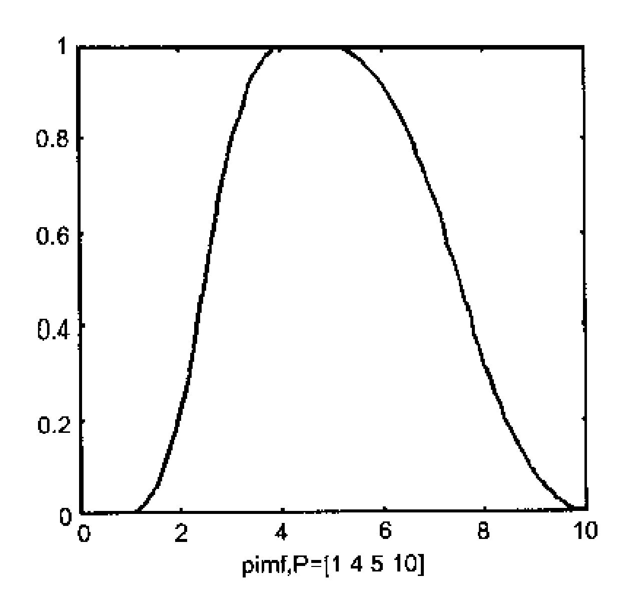


图 6-7 π型隶属度函数

6. psigmf

功能：由两个 Sigmoid 函数的积生成的新的 Sigmoid 函数。

语法：y = psigmf(x,[a1 c1 a2 c2])

说明：返回由 a1,c1,a2 和 c2 生成的两个 Sigmoid 函数之积。

【例 6】 计算两个 Sigmoid 函数之积，如图 6-8 所示。

```
x=0:0.1:10;
y=psigmf(x,[2 3 -5 8]);
plot(x,y)
xlabel('psigmf, P=[2 3 -5 8]')
```

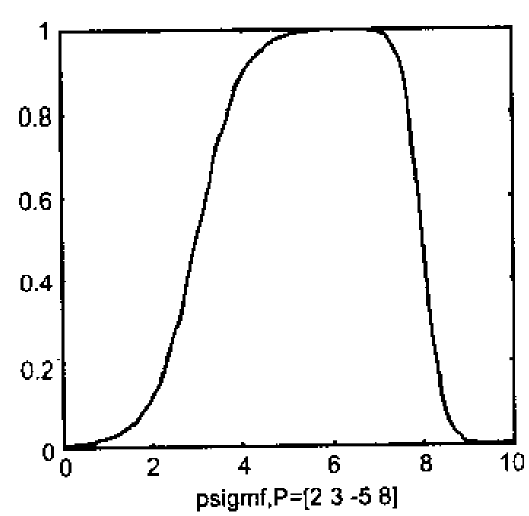


图 6-8 两个 Sigmoid 函数之积

7. smf

功能：S 型的隶属度函数。

语法：y = smf(x,[a b])

说明：该函数因为其形状与 S 相似而得名，参数 a,b 分别指定函数的两个拐点。该函数是基于样条插值的原理。

【例 7】 生成并绘制一个 S 型函数，如图 6-9 所示。

```
x=0:0.1:10;
y=smf(x,[1 8]);
plot(x,y)
xlabel('smf, P=[1 8]')
```

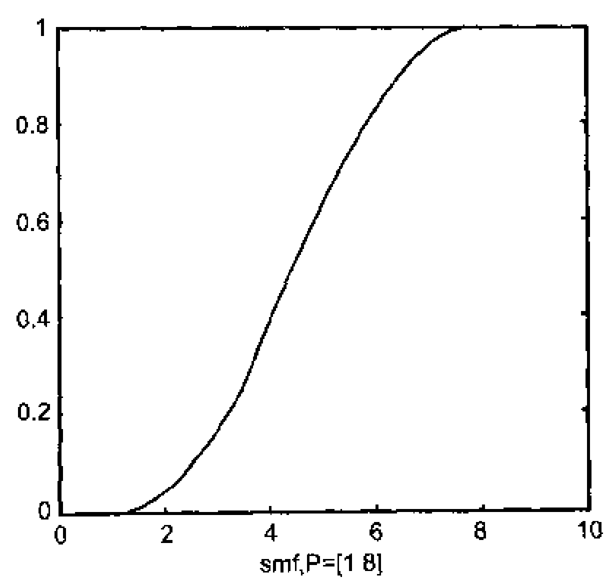


图 6-9　S 型隶属度函数

8. sigmf

功能：建立 Sigmoid 型隶属度函数。

语法：y = sigmf(x,[a c])

说明：生成由 a,c 生成的 Sigmoid 函数。

【例 8】 生成一个 Sigmoid 型隶属度函数，如图 6-10 所示。

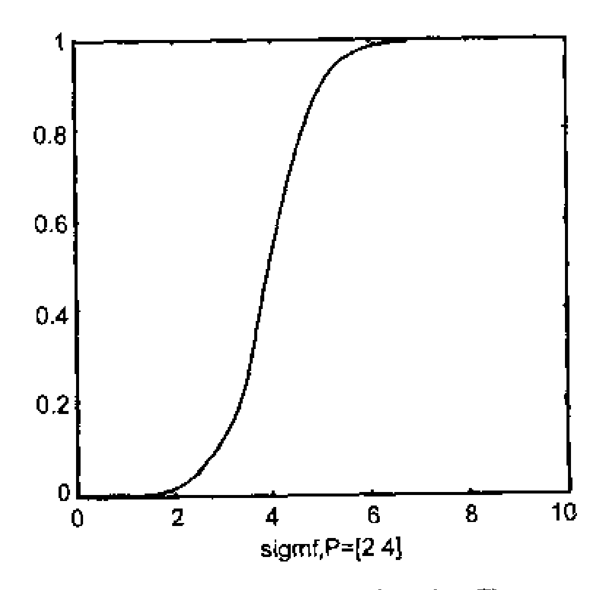


图 6-10　Sigmoid 型隶属度函数

```
x=0:0.1:10;
y=sigmf(x,[2 4]);
plot(x,y)
xlabel('sigmf, P=[2 4]')
```

9. trapmf

功能：生成梯型隶属度函数。

语法：y = trapmf(x,[a b c d])

说明：生成的梯型隶属度函数由参数 a,b,c 和 d 决定如下：

$x \leqslant a$　　$f(x) = 0$

$a \leqslant x \leqslant b$　　$f(x) = (x-a)/(b-a)$

$b \leqslant x \leqslant c$　　$f(x) = 1$

$c \leqslant x \leqslant d$　　$f(x) = (d-x)/(d-c)$

$d \leqslant x$　　0

【例 9】 生成梯型隶属度函数，如图 6-11 所示。

```
x=0:0.1:10;
y=trapmf(x,[1 5 7 8]);
plot(x,y)
xlabel('trapmf, P=[1 5 7 8]')
```

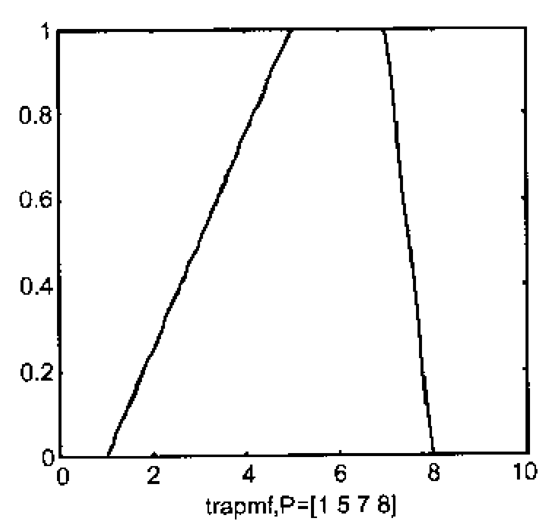


图 6-11　梯形隶属度函数

10. trimf

功能：生成三角型隶属度函数。

语法：y = trimf(x,[a b c])

说明：由参数 a,b 和 c 生成的隶属度函数如下：

$x \leqslant a$　　$f(x) = 0$

$a \leqslant x \leqslant b$　　$f(x) = (x-a)/(b-a)$

$b \leqslant x \leqslant c$　　$f(x) = (c-x)/(c-b)$

$c \leqslant x$　　0

【例 10】 生成三角型隶属度函数，如图 6-12 所示。

```
x=0:0.1:10;
y=trimf(x,[3 6 8]);
plot(x,y)
xlabel('trimf, P=[3 6 8]')
```

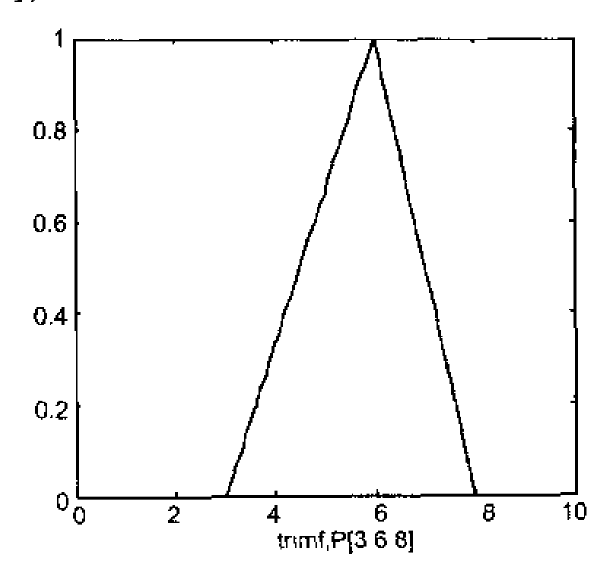


图 6-12　三角型隶属度函数

11、zmf

功能：建立 Z 型隶属度函数。

语法：y = zmf(x,[a b])

说明：该函数是因为形状类似字母 Z 而得名。参数 a,b 指定函数的两个拐点。

【例 11】 构造一Z 型隶属度函数，如图 6-13 所示。

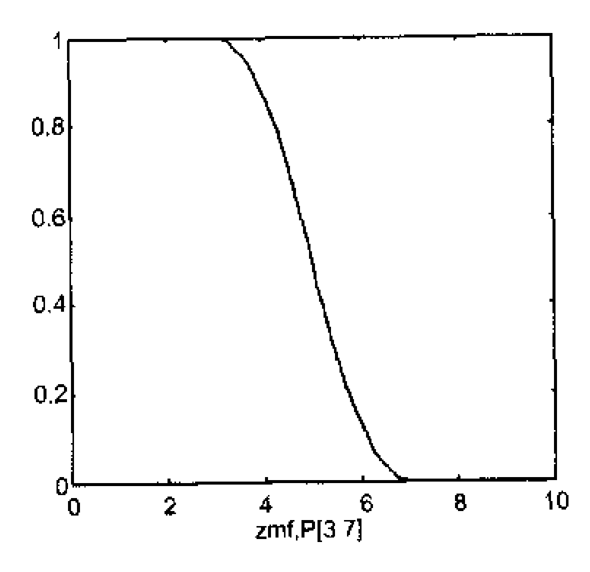


图 6-13　Z 型隶属度函数

```
x=0:0.1:10;
y=zmf(x,[3 7]);
```

```
plot(x,y)
xlabel('zmf, P=[3 7]')
```

6.2.2 模糊推理系统数据管理函数

MALTAB 模糊逻辑工具箱中提供了模糊推理系统的建立和模糊规则建立与修改的函数(表 6-2)。通过这些函数可以方便地建立和修改模糊推理系统，也可以对系统的模糊规则进行修改和解析。模糊规则的建立对模糊推理系统的建立是至关重要的，将直接影响到模糊推理系统的性能。一般常用的建立模糊规则的方法有：

- 专家系统：利用了长期从事该工作的专家或技术人员的丰富经验，因为这些经验通常本身就具有一定的模糊特性。利用这些经验可以方便地建立合理的模糊规则，由于专家对系统运行过程有深刻的认识，建立的模糊规则可以反映系统的一定特性，并且可以在实际使用中不断的改进模糊规则。
- 基于自我学习的方法：通过建立具有自我学习、修改功能的模糊控制器来自动的获得模糊规则。
- 基于过程的模糊模型：基于过程的模糊模型可以产生一组模糊控制规则来使被控制的过程达到期望的性能。

表 6-2 模糊推理系统数据管理函数

函 数 名	说 明
addmf	向模糊推理系统添加隶属度函数
addrule	向模糊推理系统添加模糊规则
addvar	向模糊推理系统添加变量
defuzz	隶属度函数的去模糊化
evalfis	执行模糊推理计算
evalmf	通用隶属度函数估计
gensurf	生成模糊推理系统的曲面并显示
getfis	获得模糊推理系统特性数据
mf2mf	隶属度函数间的参数转换
newfis	建立新的模糊推理系统
parsrule	解析模糊规则
plotfis	作图显示模糊推理系统输入输出结构
plotmf	绘制隶属度函数曲线
readfis	从磁盘载入模糊推理系统
rmmf	从模糊推理系统中删除隶属度函数
rmvar	从模糊推理系统中删除变量
setfis	设置模糊推理系统特性
showfis	显示添加了注释的模糊推理系统
showrule	显示模糊规则
writefis	将模糊推理系统保存到磁盘中

1．addmf

功能：向模糊推理系统添加隶属度函数。

语法：a = addmf(a,'varType',varIndex,'mfName','mfType',mfParams)

说明：一个隶属度函数只能添加到 MATLAB 工作空间中已经存在的模糊推理系统的语言变量中。隶属度函数按照添加的顺序来编号，第一个被添加的函数编号为 1，以后添加的依次递增。

输入参数定义为：

a：MATLAB 模糊推理系统的变量名；

'varType'：指定该变量的类型（'input'或'output'）；

varIndex：指定该变量的编号；

'mfName'：指定隶属度函数的名字；

'mfType'：指定隶属度函数的类型；

mfParams：指定隶属度函数的参数。

【例 1】 利用 addmf 添加一个隶属度函数，该函数曲线如图 6-14 所示。

```
a=newfis('TEST');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'a1','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'b1','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'c1','gaussmf',[1.5 10]);
plotmf(a,'input',1)
```

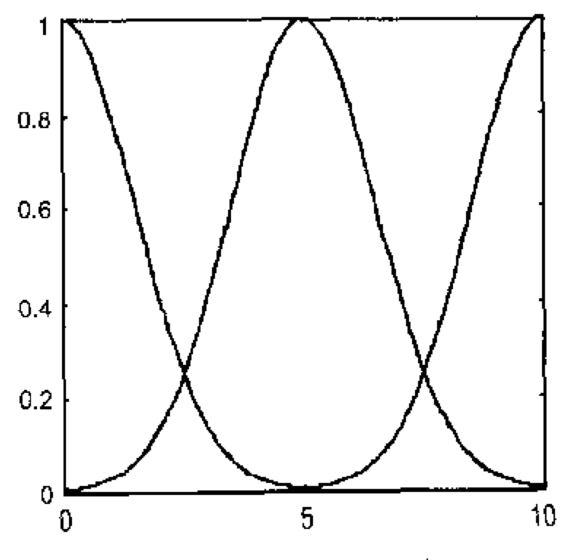


图 6-14　隶属度函数曲线

2．addrule

功能：向模糊推理系统添加模糊规则。

语法：a = addrule(a,ruleList)

说明：输入参数 a 为模糊推理系统，ruleList 以向量的形式给出模糊规则。该向量以严格的形式给出模糊规则。如果模糊推理系统有 m 个输入，n 个输出，则该向量有 m+n+2 列。

前 m 列表示系统输入，每个数值指明该变量的隶属度函数的序号；

接着的 n 列表示系统输出，每个数值指明该变量的隶属度函数的序号；

第 m+n+1 列指出该规则的权重，通常为 1；

第 m+n+2 列为 0 或 1。如果为 0，则表示模糊规则间是“或”的关系；如果为 1，则表示模糊规则间是“与”的关系。

【例 2】

```
ruleList=[ 1 1 1 1 1
           1 2 2 1 1];
a = addrule(a,ruleList);
```

3. addvar

功能：向模糊推理系统添加变量。

语法：a = addvar(a,'varType','varName',varBounds)

说明：输入参数：

a：模糊推理系统；

'varType'：变量的类型；

'varName'：变量的名称；

varBounds：指定变量的取值范围。

对于添加到同一模糊推理系统中的变量，将按照添加的顺序来自动安排变量的编号，从 1 开始，往后递增。对输入、输出变量则独立地开始编号。

【例 3】

```
a=newfis('tipper');
a=addvar(a,'input','sample',[0 10]);
getfis(a,'input',1)
```

MATLAB 返回：

```
Name = sample
NumMFs = 0
MFLabels =
Range = [0 10]
```

4. defuzz

功能：隶属度函数的去模糊化。

语法：out = defuzz(x,mf,type)

说明：输入参数说明：

x：变量的名字；

mf：待去模糊化的隶属度函数；

type：去模糊化的方法，它可以取以下的 5 个值之一：

- centroid：面积中心法；
- bisector：面积平分法；
- mom：平均最大隶属度法；
- som：最大隶属度取最小法；
- lom：最大隶属度取最大法。



【例 4】

```
x = -10:0.1:10;
mf = trapmf(x,[-10 -8 -4 7]);
xx = defuzz(x,mf,'centroid')
```

5. evalfis

功能：执行模糊推理计算。

语法：output= evalfis(input,fismat)

output= evalfis(input,fismat, numPts)

[output, IRR, ORR, ARR]= evalfis(input,fismat)

[output, IRR, ORR, ARR]= evalfis(input,fismat, numPts)

说明：输入参数定义：

input：输入构成的模糊矩阵；

fismat：模糊推理系统名；

numPts：指明用于采样的点数，如果不指明则用缺省的 101 点采样。

输出参数定义：

output：输出 M*L 的矩阵，M 为输入变量的个数，L 为模糊推理系统输出的变量个数；

IRR：使用隶属度函数估计输入变量的结果，为 numRules*N 维的矩阵，numRules 为模糊规则的个数，N 为输入变量的个数；

ORR：使用隶属度函数估计输出变量的结果，为 numPts* (numRules*L)维的矩阵，numRules 为模糊规则的个数，L 为输出变量的个数；

ARR：numPts*L 维的矩阵，包含每个输入在每个采样点处的值。

【例 5】

```
fismat = readfis('tipper');
out = evalfis([2 1; 4 9],fismat)
```

6. evalmf

功能：通用隶属度函数估计。

语法：y = evalmf(x,mfParams,mfType)

说明：输入参数 x 为隶属度函数的变量，mfType 为工具箱中隶属度函数名，mfParams 为该函数的参数列表。

【例 6】

```
x=0:0.1:10;
mfparams = [2 4 6];
mftype = 'gbellmf';
y=evalmf(x,mfparams,mftype);
plot(x,y)
xlabel('gbellmf, P=[2 4 6]')
```

生成图形如图 6-15 所示。

7. gensurf

功能：生成模糊推理系统的曲面并显示。

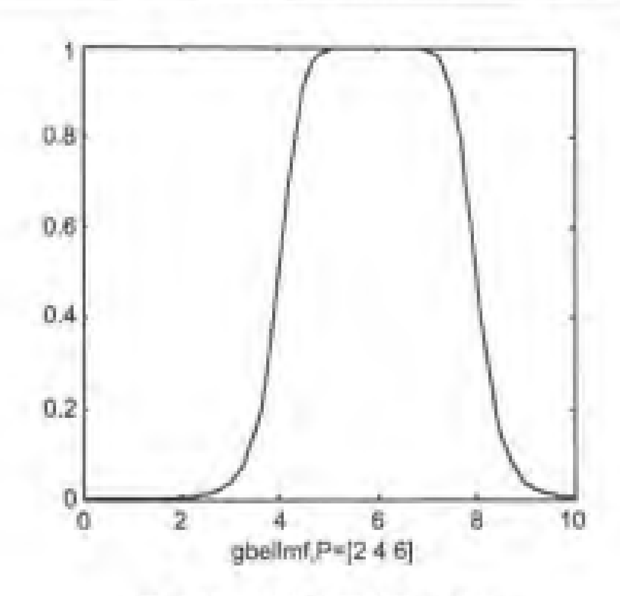


图 6-15 钟型隶属度函数估计

语法：gensurf(fis)

gensurf(fis,inputs,output)

gensurf(fis,inputs,output,grids)

gensurf(fis,inputs,output,grids,refinput)

说明：输入参数定义：

fis：为模糊推理系统；

inputs：模糊推理系统的输入变量；

output：模糊推理系统的输出变量；

grids：可选参数，表示 X 和 Y 方向的网格数目；

refinput：指定保持不变的输入变量。

【例 7】 绘制模糊推理系统特性曲面，如图 6-16 所示。

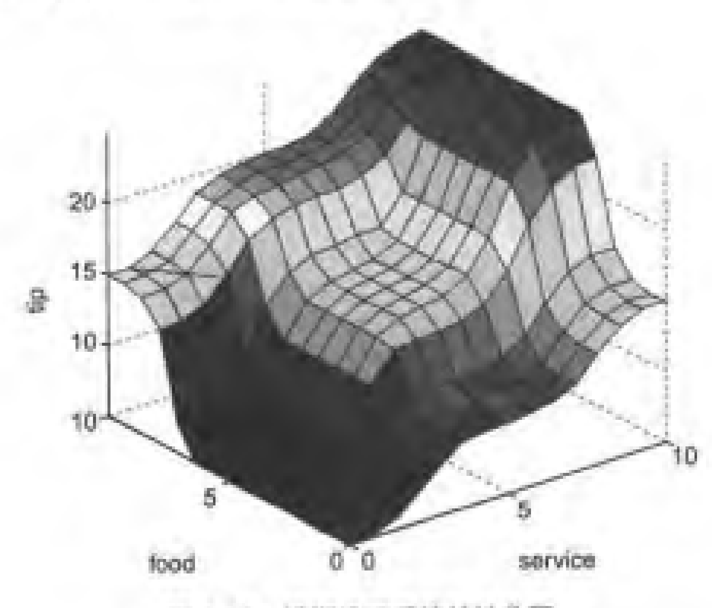


图 6-16 模糊推理系统特性曲面

```
a = readfis('tipper');
gensurf(a)
```

8. getfis

功能：获得模糊推理系统特性数据。

语法：getfis(a)

getfis(a,'fisprop')

getfis(a,'vartype',varindex,'varprop')

getfis(a,'vartype',varindex,'mf',mfindex)

getfis(a,'vartype',varindex,'mf',mfindex,'mfprop')

说明：输入参数定义为：

a：模糊推理系统；

'fisprop'：字符串，用来指明需要获得哪个字段；

'vartype'：字符串，用来指明需要获得哪个属性的字段，如 input 或 output；

varindex：用来指定需要获取的变量的编号；

'varprop'：字符串，用来指明需要获取变量的属性值；

'mf'：获取隶属度函数的函数名；

mfindex：获取隶属度函数的编号。

【例 8】

(1) 使用单参数调用：

```
a = readfis('tipper');
getfis(a)
```

MATLAB 返回：

```
Name          = tipper
        Type          = mamdani
        NumInputs = 2
        InLabels   =
           service
           food
        NumOutputs = 1
        OutLabels =
           tip
        NumRules = 3
        AndMethod = min
        OrMethod = max
        ImpMethod = min
        AggMethod = max
DefuzzMethod = centroid
```

(2) 如果使用两个参数调用：

```
getfis(a,'type')
```

MATLAB 系统返回：

```
out =
mamdani
ans =
mamdani
```

(3) 如果使用三个参数调用：

```
getfis(a,'input',1)
```

MATLAB 返回：

```
Name = service
NumMFs = 3
MFLabels =
        poor
        good
        excellent
Range = [0 10]
```

9．mf2mf

功能：隶属度函数间的参数转换。

语法：outParams = mf2mf(inParams,inType,outType)

说明：输入参数说明：inParams 输入参数；

inType：输入隶属度函数的类型；

outType：输出隶属度函数的类型。

【例 9】 将钟型隶属度函数转换为三角形隶属度函数，如图 6-17 所示。

```
x=0: 0.1: 5;
mfp1 = [1 2 3];
mfp2 = mf2mf(mfp1,'gbellmf','trimf');
plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

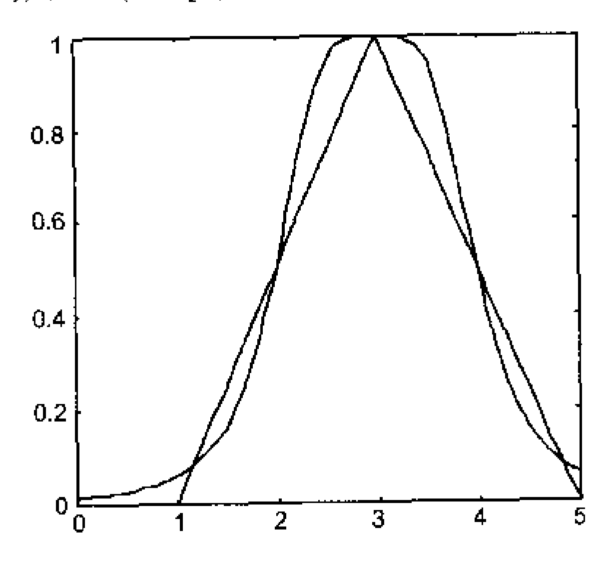


图 6-17 隶属度函数间的转换

10．newfis

功能：建立新的模糊推理系统。

语法：a=newfis(fisName,fisType,andMethod,orMethod,impMethod, aggMethod,defuzzMethod)

说明：用于创建一个新的模糊推理系统。

输入参数定义为：

fisName：模糊推理系统的名称；

fisType：模糊推理系统的类型；

andMethod："与" 运算符；

orMethod："或" 运算符；

impMethod：模糊蕴涵方法；

aggMethod：各条规则推理结果的综合方法；

defuzzMethod：去模糊化方法。

【例 10】 生成一个模糊推理系统。

```
a=newfis('newsys');
getfis(a)
```

MATLAB 返回：

```
    Name        = newsys
    Type        = mamdani
    NumInputs = 0
    InLabels    =
    NumOutputs = 0
    OutLabels =
    NumRules = 0
    AndMethod = min
    OrMethod = max
    ImpMethod = min
    AggMethod = max
    DefuzzMethod = centroid
ans =
newsys
```

11．parsrule

功能：解析模糊规则。

语法：fis2 = parsrule(fis,txtRuleList)

fis2 = parsrule(fis,txtRuleList,ruleFormat)

fis2 = parsrule(fis,txtRuleList,ruleFormat,lang)

说明：该函数解析模糊推理系统中模糊语言规则。

fis：模糊推理系统；

txtRuleList：模糊语言规则；

ruleFormat：规则的格式，包括：'verbose'、'symbolic'、'indexed'；

lang：当使用该可选参数时，ruleFormat 选用'verbose'，关键词存储在 lang 中，语言必须为'english'、'francais'或'deutsch'之一。

【例 11】

```
a = readfis('tipper');
ruleTxt = 'if service is poor then tip is generous';
a2 = parsrule(a,ruleTxt,'verbose');
showrule(a2)
```

MATLAB 返回：

```
ans =
1. If (service is poor) then (tip is generous) (1)
```

12. plotfis

功能：作图显示模糊推理系统输入输出结构。

语法：plotfis(fismat)

说明：该函数在左边显示模糊推理系统的输入和隶属度函数，在右边显示模糊推理系统的输出。

【例 12】 作图显示模糊推理系统 tipper 的结构，如图 6-18 所示。

```
a = readfis('tipper');
plotfis(a)
```

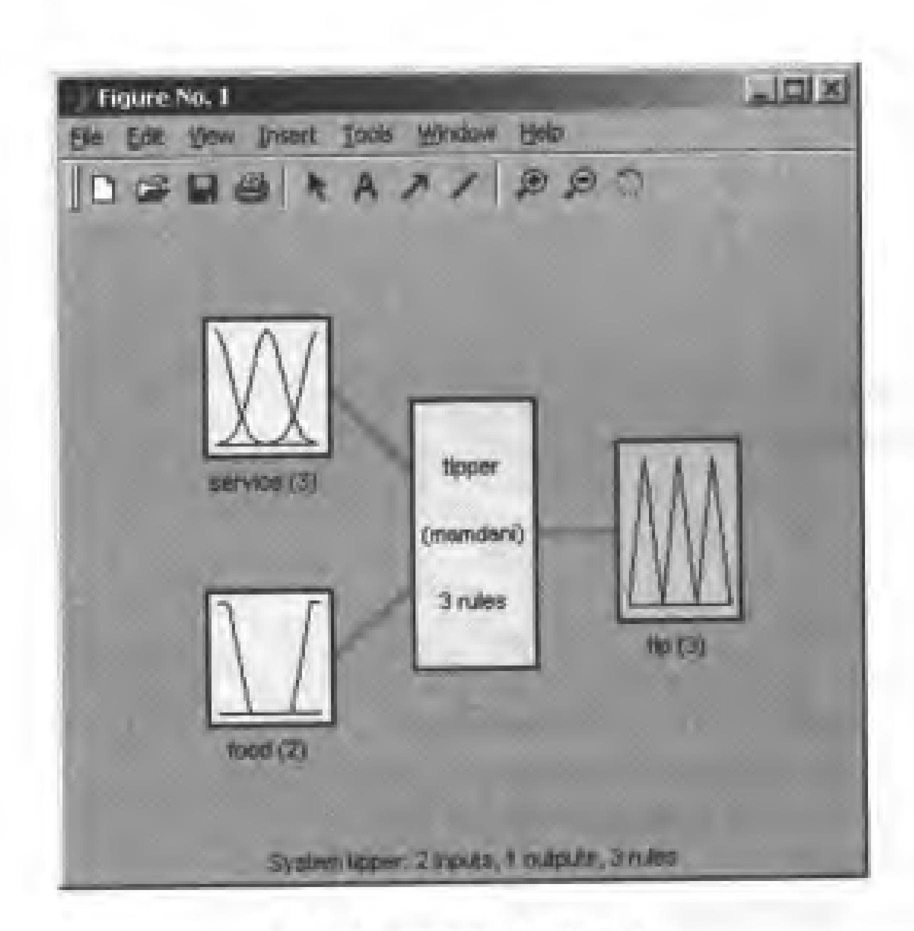


图 6-18 模糊推理系统结构图

13. plotmf

功能：绘制隶属度函数曲线

语法：plotmf(fismat,varType,varIndex)

说明：绘制隶属度函数图形，输入参数说明：

fismat：模糊推理系统；

varType：变量的类型；

varIndex：变量的编号。

【例 13】 绘制模糊推理系统 tipper 中的隶属度函数，如图 6-19 所示。

```
a = readfis('tipper');
plotmf(a,'input',1)
```

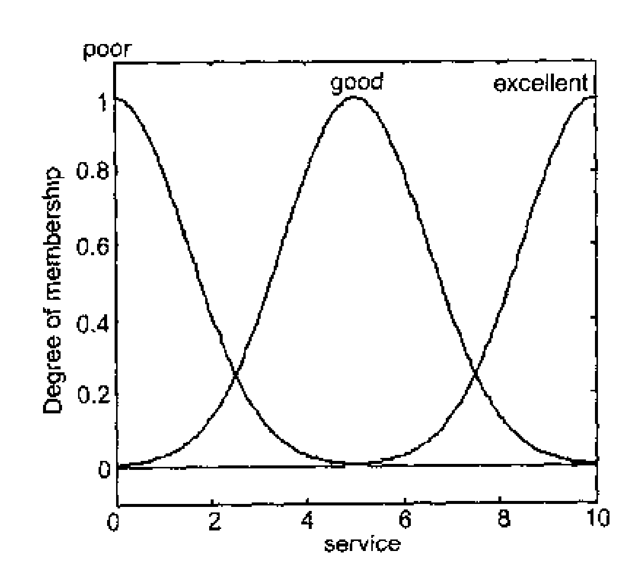


图 6-19 隶属度函数曲线

14. readfis

功能：从磁盘载入模糊推理系统。

语法：fismat = readfis('filename')

说明：从'filename'中读取模糊推理系统。

【例 14】

```
fismat = readfis('tipper');
getfis(fismat)
```

MATLAB 返回：

```
Name = tipper
Type = mamdani
NumInputs = 2
InLabels =
   service
   food
```

```
NumOutputs = 1
OutLabels =
tip
NumRules = 3
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans =
tipper
```

15. rmmf

功能：从模糊推理系统中删除隶属度函数。

语法：fis = rmmf(fis,'varType',varIndex,'mf',mfIndex)

说明：输入参数说明：

fis：模糊推理系统；

'varType'：变量的类型；

varIndex：变量的编号；

'mf'：隶属度函数的名字；

mfIndex：隶属度函数的编号。

【例 15】

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
a = addmf(a,'input',1,'cold','trimf',[0 30 60]);
getfis(a,'input',1)
```

MATLAB 返回为：

```
Name = temperature
NumMFs = 1
MFLabels =
    cold
Range = [0 100]
```

若采用如下语句：

```
b = rmmf(a,'input',1,'mf',1);
getfis(b,'input',1)
```

则 MATLAB 返回：

```
Name = temperature
NumMFs = 0
MFLabels =
Range = [0 100]
```




16. rmvar

功能：从模糊推理系统中删除变量。

语法：[fis2,errorStr] = rmvar(fis,'varType',varIndex)

fis2 = rmvar(fis,'varType',varIndex)

说明：输入参数说明：

fis：模糊推理系统；

'varType'：变量类型；

varIndex：变量编号。

输出参数说明：

fis2 为删除了变量以后的模糊推理系统；

errorStr：为系统返回的任何出错信息。

【例 16】

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
getfis(a)
```

此时 MATLAB 返回：

```
Name = mysys
Type        = mamdani
NumInputs = 1
InLabels  =
   temperature
NumOutputs = 0
OutLabels =
NumRules = 0
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans =
mysys
```

使用该函数删除变量的语句为：

```
b = rmvar(a,'input',1);
getfis(b)
```

此时 MATLAB 返回为：

```
Name = mysys
Type = mamdani
NumInputs = 0
InLabels =
```

```
NumOutputs = 0
OutLabels =
NumRules = 0
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans =
mysys
```

17. setfis

功能：设置模糊推理系统特性。

语法：a = setfis(a,'fispropname','newfisprop')

a = setfis(a,'vartype',varindex,'varpropname','newvarprop')

a = setfis(a,'vartype',varindex,'mf',mfindex,'mfpropname','newmfprop')

说明：a = setfis(a,'fispropname','newfisprop')用于设定模糊推理系统的全局属性；它包括：

- name：模糊推理系统的名称；
- type：模糊推理系统的类型；
- andmethod：“与”运算方法；
- ormethod：“或”运算方法；
- impmethod：模糊蕴涵方法；
- aggmethod：各个规则推理结果的综合方法；
- defuzzmethod：去模糊化方法。

a = setfis(a,'vartype',varindex,'varpropname','newvarprop')用于设定模糊推理系统某一变量的属性，包括：name 和 bounds 属性。

a = setfis(a,'vartype',varindex,'mf',mfindex,'mfpropname','newmfprop')用于设定某一隶属度函数的属性，包括：name、type 和 params 属性。

【例 17】

(1) 三参数调用时：

```
a = readfis('tipper');
a2 = setfis(a, 'name', 'eating');
getfis(a2, 'name');
```

MATLAB 返回：

```
out =
eating
```

(2) 五参数调用时：

```
a2 = setfis(a,'input',1,'name','help');
getfis(a2,'input',1,'name')
```

MATLAB 返回：

```
ans =
    help
```

(3) 七参数调用时：

```
a2 = setfis(a,'input',1,'mf',2,'name','wretched');
getfis(a2,'input',1,'mf',2,'name')
```

MATLAB 返回：

```
ans =
    wretched
```

18. showfis

功能：显示添加了注释的模糊推理系统。

语法：showfis(fismat)

说明：fismat 为模糊推理系统。

【例 18】 a = readfis('tipper');

```
showfis(a)
```

MATLAB 返回：

```
1.  Name            tipper
2.  Type            mamdani
3.  Inputs/Outputs  [2 1]
4.  NumInputMFs     [3 2]
5.  NumOutputMFs    3
6.  NumRules        3
7.  AndMethod       min
8.  OrMethod        max
9.  ImpMethod       min
10. AggMethod       max
11. DefuzzMethod    centroid
12. InLabels        service
13.                 food
14. OutLabels       tip
15. InRange         [0 10]
16.                 [0 10]
17. OutRange        [0 30]
18. InMFLabels      poor
19.                 good
20.                 excellent
21.                 rancid
22.                 delicious
23. OutMFLabels     cheap
```

24. average
25. generous
26. InMFTypes gaussmf
27. gaussmf
28. gaussmf
29. trapmf
30. trapmf
31. OutMFTypes trimf
32. trimf
33. trimf
34. InMFParams [1.5 0 0 0]
35. [1.5 5 0 0]
36. [1.5 10 0 0]
37. [0 0 1 3]
38. [7 9 10 10]
39. OutMFParams [0 5 10 0]
40. [10 15 20 0]
41. [20 25 30 0]
42. Rule Antecedent [1 1]
43. [2 0]
44. [3 2]
42. Rule Consequent 1
43. 2
44. 3
42. Rule Weigth 1
43. 1
44. 1
42. Rule Connection 2
43. 1
44. 2

19. showrule

功能：显示模糊规则。

语法：showrule(fis)
showrule(fis,indexList)
showrule(fis,indexList,format)
showrule(fis,indexList,format,Lang)

说明：输入参数 fis 为模糊推理系统。

indexList：所需要显示的模糊规则列表；

format：规定模糊规则以什么方式显示，可以使用如下三种方式之一：详述式（verbose）、

符号式（symbolic）和编号式（indexed）。

Lang：语言选择，可以为'english'、'francais'或'deutsch'。

【例 19】

```
a = readfis('tipper');
showrule(a,1)
ans =
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
showrule(a,2)
ans =
2. If (service is good) then (tip is average) (1)
showrule(a,[3 1],'symbolic')
ans =
3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1. (service==poor) | (food==rancid) => (tip=cheap) (1)
showrule(a,1:3,'indexed')
ans =
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2
```

20．writefis

功能：将模糊推理系统保存到磁盘中。

语法：writefis(fismat)

writefis(fismat,'filename')

writefis(fismat,'filename','dialog')

说明：writefis(fismat)生成一个文件对话框窗口，提示用户输入文件名；writefis(fismat,'filename')直接用'filename'作文件名存盘；writefis(fismat,'filename','dialog')打开一个以'filename'为缺省文件名的文件对话框。

【例 20】

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
```

6.3　逻辑工具箱的图形用户界面

MALTAB 模糊逻辑工具箱为模糊推理系统的管理提供了一个图形化的用户界面，通过该

界面可以方便直观地修改和管理模糊推理系统。命令见表 6-3。

表 6-3　图形用户界面函数列表

函 数 名	说　　明
anfisedit	神经模糊推理系统建模的图形用户界面工具
fuzzy	基本模糊推理系统编辑器
mfedit	隶属度函数编辑器
ruleedit	模糊推理规则编辑器
ruleview	模糊推理规则观察器
surfview	模糊推理输出特性曲面观察器

1．anfisedit

功能：神经模糊推理系统建模的图形用户界面工具。

语法：anfisedit('a')

anfisedit(a)

anfisedit

说明：anfisedit 打开神经模糊推理系统的图形用户界面。

anfisedit('a')打开磁盘中文件名为 a.fis 的神经模糊推理系统。

anfisedit(a)打开系统神经模糊推理系统变量。

使用无参数命令打开神经模糊推理系统建模的图形用户界面，如图 6-20 所示。

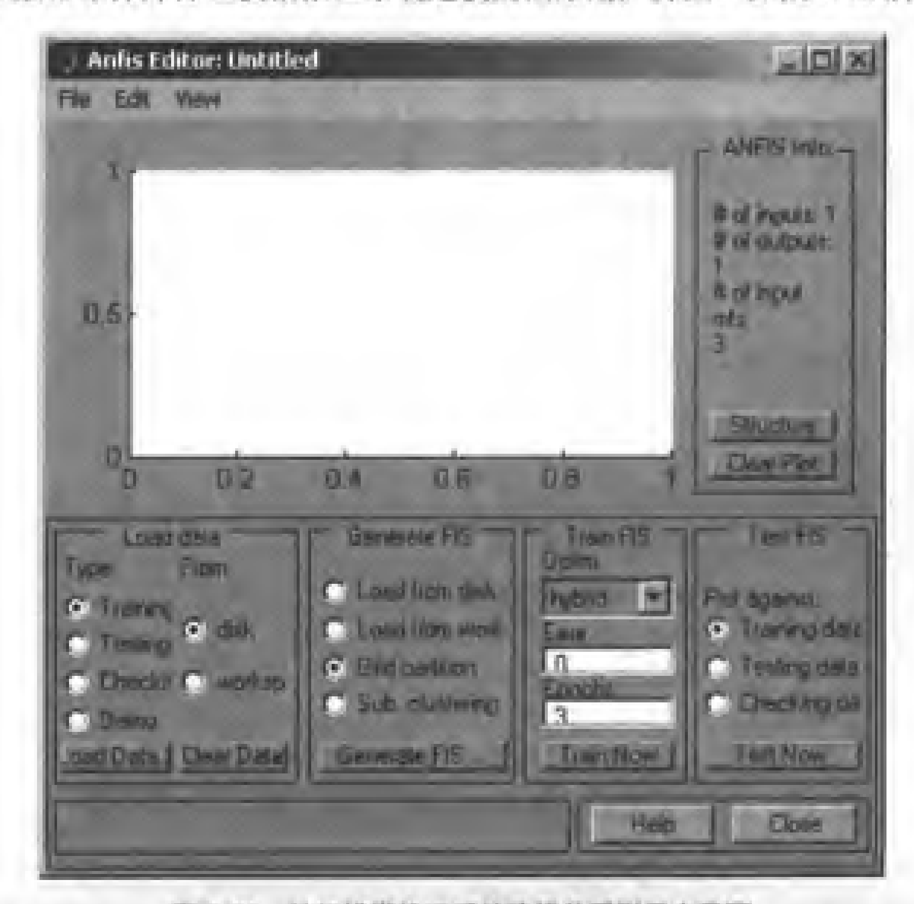


图 6-20　神经模糊推理系统建模的图形用户界面

通过图 6-20 所示的图形用户界面中的菜单条，可以打开或保存系统等操作。

View 菜单下的条目说明如下：

Edit FIS properties：打开 FIS 编辑器；

Edit rules ：打开模糊规则编辑器；

Edit membership functions 打开隶属度函数编辑器；

View rules ：打开模糊规则观测器；

View surface ：打开特性曲面观测器。

通过界面上的检查框可以选择加载数据的类型，输入训练数据、测试数据、演示数据等。加载了演示数据后的窗口如图 6-21 所示。

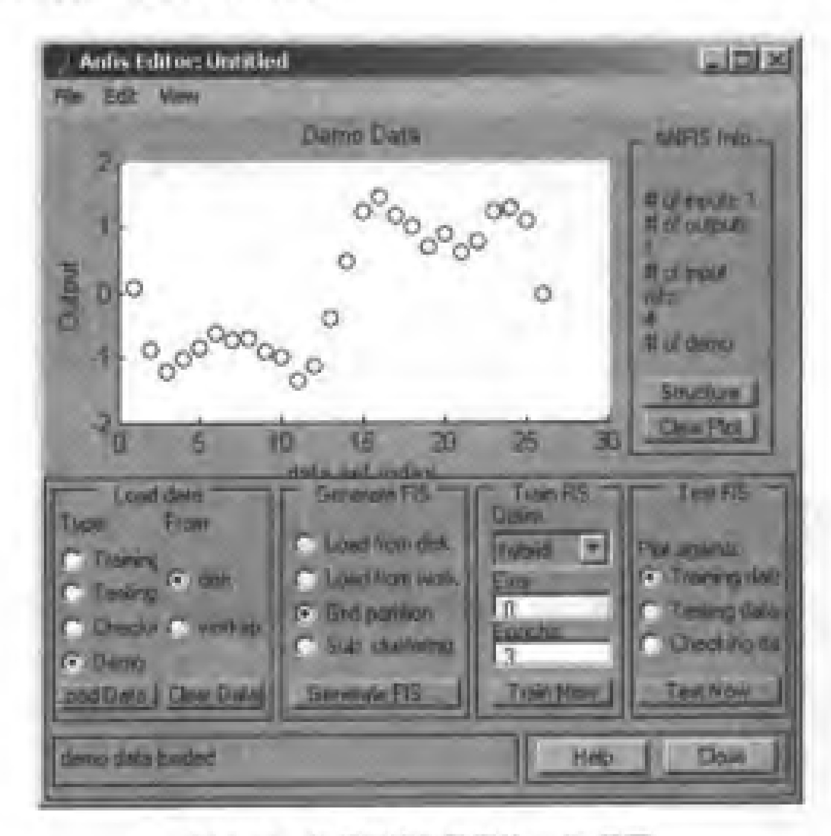


图 6-21　加载了演示数据的 anfis 界面

在生成模糊推理系统时，可以通过检查框选择模糊推理系统的生成方法，如网格分隔法、减法聚类法等。当单击生成模糊推理系统时，系统会弹出一个对话框，要求输入模糊推理系统的有关信息，如图 6-22 所示。

在进行神经模糊推理训练前，可以指定优化方法及优化的参数。对于加载的演示数据，在进行 anfis 训练后，窗口显示了优化过程中误差变化的情况。如图 6-23 所示。

在训练完成后可以方便地得到模糊神经网络的结构，如图 6-24 所示。也可以将 ANFIS 的输出数据与测试数据比较，结果如图 6-25 所示。

2．fuzzy

功能：基本模糊推理系统编辑器。

语法：fuzzy

fuzzy(fismat)

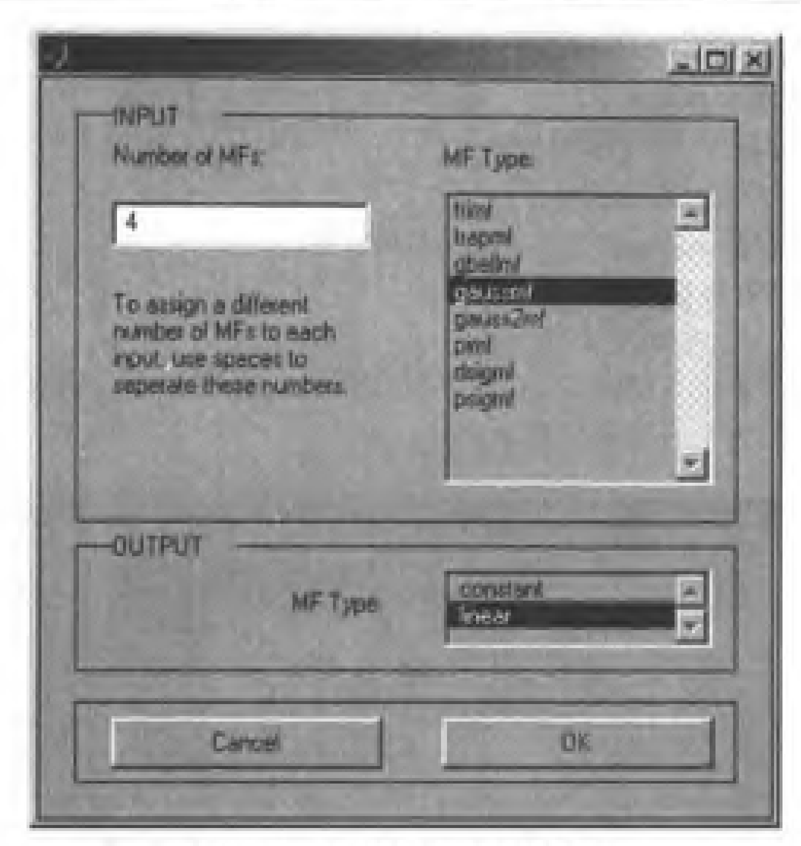


图 6-22　生成模糊推理系统对话框

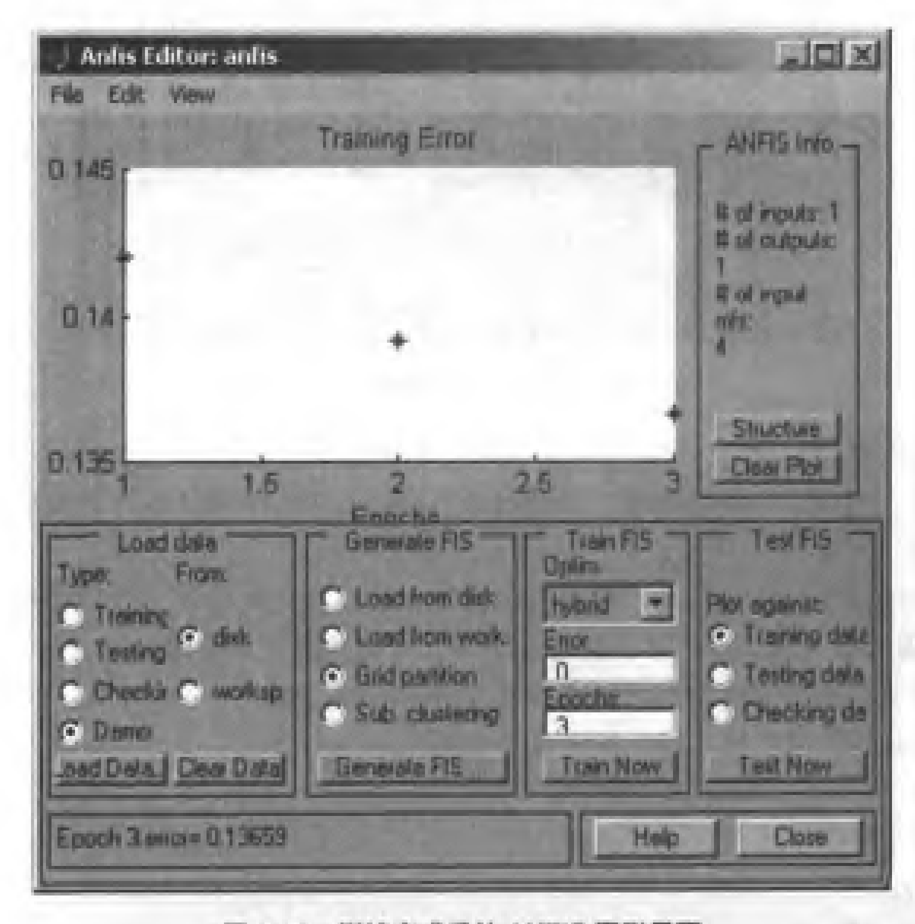


图 6-23　训练完成后的 ANFIS 图形界面

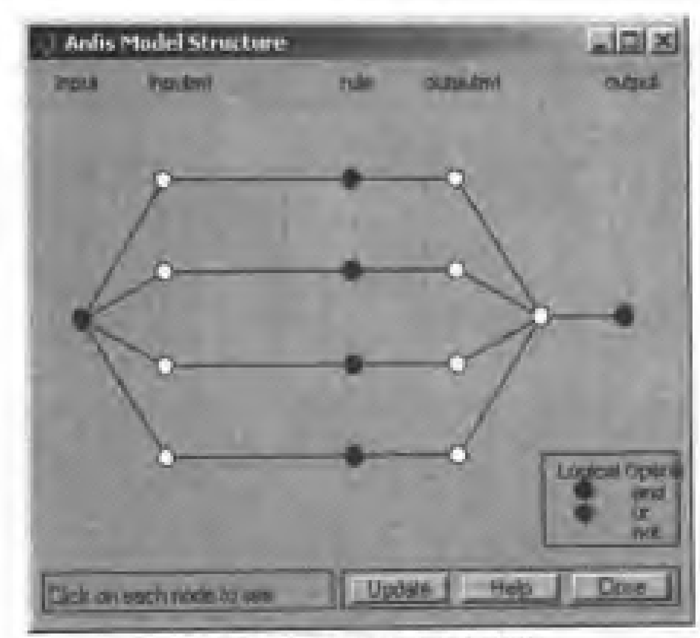


图 6-24　模糊神经网络的结构

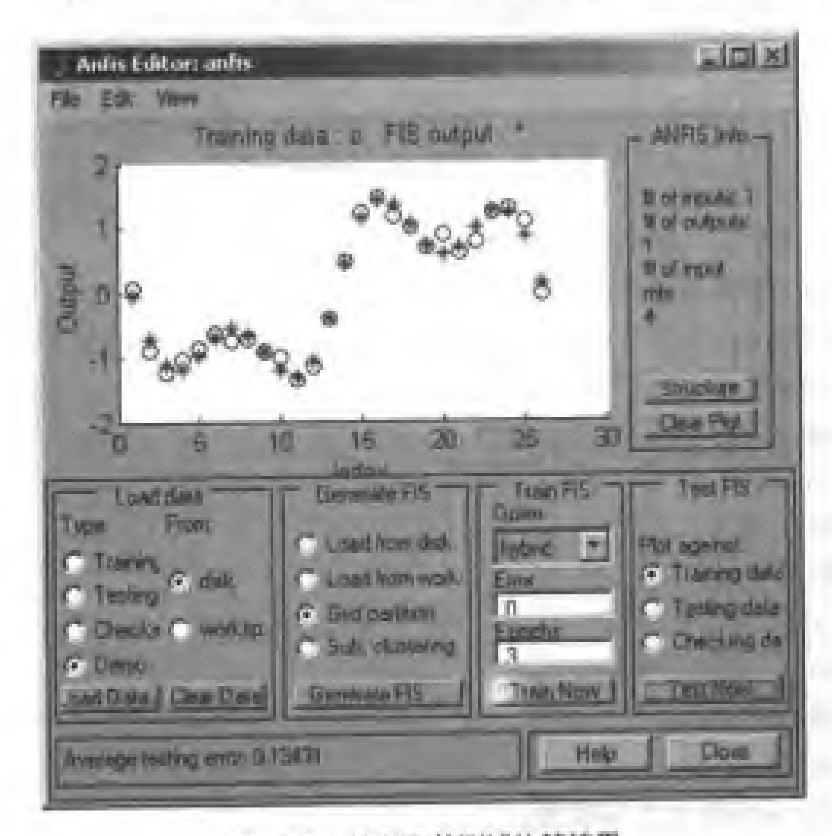


图 6-25　ANFIS 的测试比较结果

说明：打开的模糊推理系统编辑器如图 6-26 所示。

菜单栏说明：

(1) 文件（File）菜单：

New Mamdani FIS：新建 Mamdani 型模糊推理系统；

New Sugeno FIS：新建 Sugeno 型模糊推理系统；

Open from disk：打开一个模糊推理系统文件；

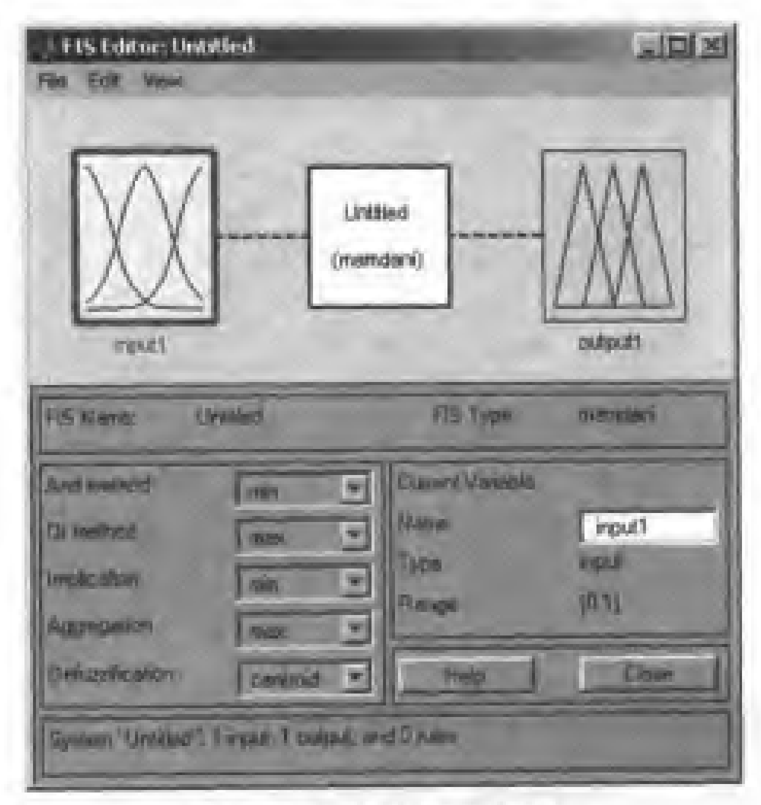


图 6-26　模糊推理系统编辑器

Save to disk：保存一个模糊推理系统文件；
Save to disk as：模糊推理系统另存为文件；
Open from workspace：从当前的 workspace 打开一个模糊推理系统；
Save to workspace：保存到当前工作中；
Save to workspace as：另存到工作空间；
Close window：关闭窗口。
(2) 编辑（Edit）菜单：
Add input：添加输入变量；
Add output：添加输出变量；
Remove variable：删除变量；
Undo：撤消最近的操作。
(3) 视图（View）菜单：
Edit MFs：打开隶属度函数编辑器；
Edit rules：打开模糊规则编辑器；
Edit anfis：打开模糊推理系统属性修改器；
View rules：打开模糊规则观察器；
View surface：打开特性曲面观察器。

3．mfedit

功能：隶属度函数编辑器。
语法：mfedit('a')

mfedit(a)

mfedit

说明：mfedit(a)打开当前工作空间中的变量 a；

mfedit('a')打开文件 a.fis；

mfedit 打开一个空白的隶属度函数编辑器。

使用 mfedit('tipper')打开隶属度函数编辑器如图 6-27 所示。

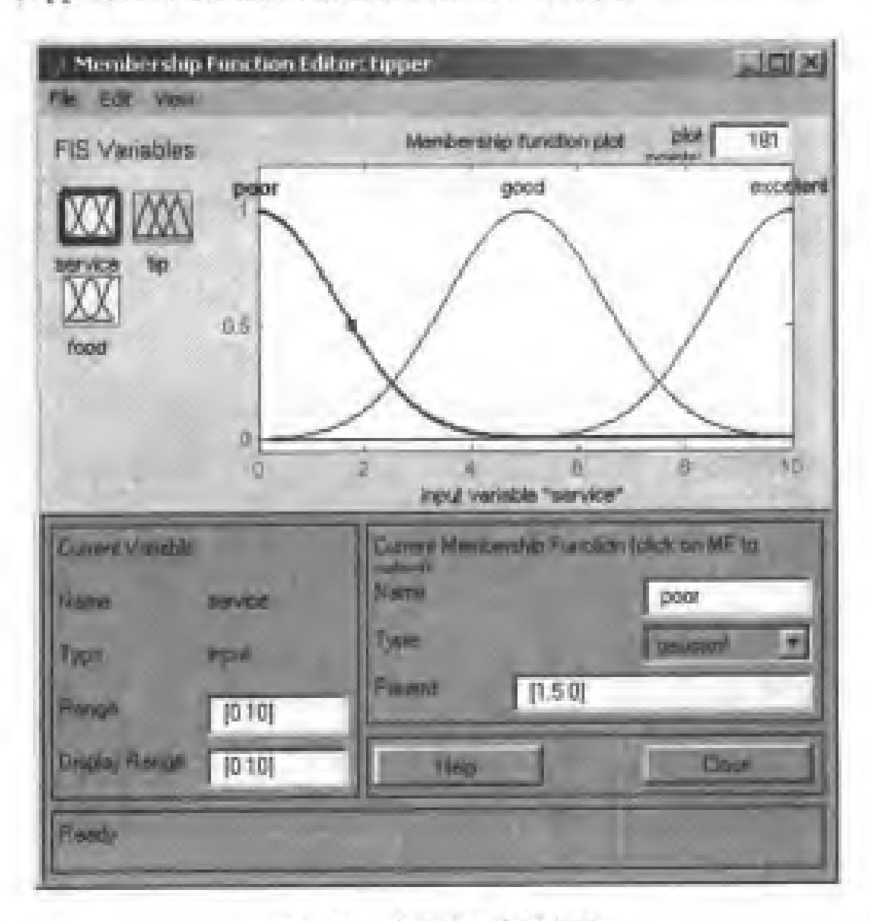


图 6-27　隶属度函数编辑器

菜单栏各项说明如下：

(1) 编辑（Edit）菜单：

Add MF：添加隶属度函数；

Add custom MF：添加定制的隶属度函数；

Remove current MF：删除当前隶属度函数；

Remove all MFs：删除当前模糊推理系统变量中的所有隶属度函数；

Undo：撤消最近的操作。

(2) 视图（View）菜单：

Edit FIS properties：打开 FIS 编辑器；

Edit rules：打开模糊规则编辑器；

View rules：打开模糊规则观测器；

View surface：打开特性曲面观测器。

4. ruleedit

功能：模糊推理规则编辑器。

语法：ruleedit('a')

ruleedit(a)

说明：通过变量或文件打开模糊推理规则编辑器。使用 ruleedit('tipper')打开模糊推理规则编辑器如图 6-28 所示。



图 6-28 模糊规则编辑器

菜单栏说明：

(1) 编辑（Edit）菜单：

Undo：撤消最近的操作；

Edit FIS properties：编辑模糊推理系统的属性；

Edit membership functions：编辑隶属度函数。

(2) 视图（View）菜单：

View rules：打开模糊规则观察器；

View surface：打开特性曲面观测器。

(3) 选项（Options）菜单：

Language：选择使用的语言；

Format：选择使用模糊规则的方式：

- verbose：语言型；
- symbolic：符号型；
- indexed：索引型。



5. ruleview

功能：模糊推理规则观察器。

语法：ruleview('a')

说明：打开磁盘中存储的模糊推理系统，并显示模糊推理规则。使用 ruleview('tipper')打开模糊规则观察器，如图 6-29 所示。

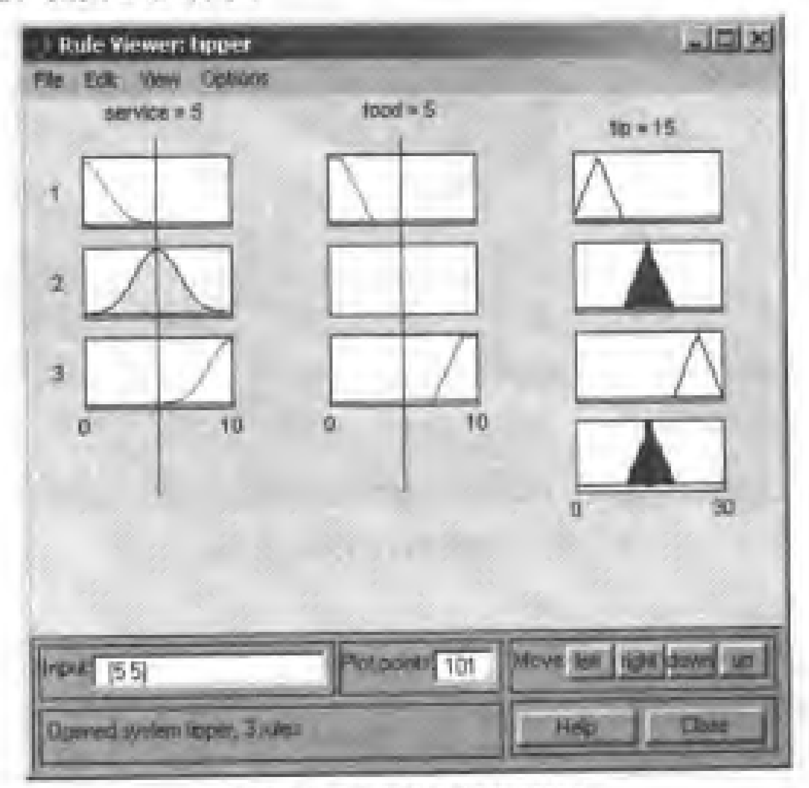


图 6-29　模糊推理规则观察器

菜单栏说明：

(1) 编辑（Edit）菜单：

Undo：撤消最近的操作；

Edit FIS properties：编辑模糊推理系统的属性；

Edit membership functions：编辑隶属度函数；

Edit rules：打开模糊规则编辑器；

(2) 选项（Options）菜单：

Format：选择使用模糊规则的方式：

- verbose：语言型；
- symbolic：符号型；
- indexed：索引型。

6. surfview

功能：模糊推理输出特性曲面观察器。

语法：surfview('a')

说明：打开磁盘中存储的模糊推理系统，并显示特性曲面。使用 surfview ('tipper')打开模糊推理输出特性曲面观察器，如图 6-30 所示。

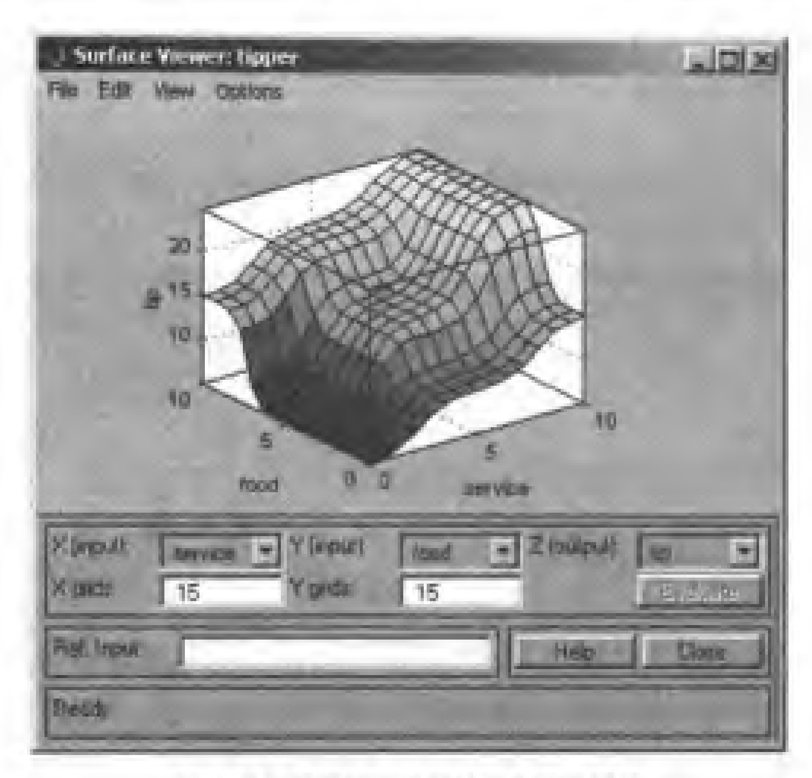


图 6-30　模糊推理输出特性曲面观察器

菜单栏说明：

(1) 编辑（Edit）菜单：

Undo：撤消最近的操作；

Edit FIS properties：编辑模糊推理系统的属性；

Edit membership functions：编辑隶属度函数；

Edit rules：打开模糊规则编辑器；

View rules：打开模糊规则观察器。

(2) 选项（Options）菜单：

Plot：选择作图的各项参数；

Color Map：作图的颜色选择；

Always evaluate：选中表示，在对模糊推理系统作出任何改动（例如，改变作图的网格点）后，自动地对特性曲面重新绘制。

6.4　模糊推理系统的高级应用

在实际中常常需要将一些事物按一定的标准进行分类，例如在地质勘探中，要根据矿石标本的物探、化探等指标数据对标本进行分类。在医疗诊断中，要根据患者的各种化验指标数据，对患者所患疾病进行分类等等；对所研究的事物按一定标准进行分类的数学方法称为聚类分析。

由于在实际事物之间在很多方面并没有一个截然的区别界限，事物各指标的变化常常具有连续性，所以分类本身就带有模糊性。可见用模糊数学方法解决聚类问题就会更加符合实



际。模糊聚类方法给出的结论不是说事物绝对地属于一类或绝对地不属于一类，而是在什么程度上属于一类，在什么程度上不属于一类。模糊聚类方法不止一种，MATLAB 模糊逻辑工具箱中提供了两种聚类方法的支持：一种是 C-均值聚类方法，另一种是减聚类方法。函数 fcm 和 subclust 就是 MALTAB 中提供的两个模糊聚类函数。说明见表 6-4。

表 6-4　模糊推理系统的高级应用

函 数 名	说　　明
anfis	利用 Sugeno 型模糊推理系统建模
fcm	利用模糊 C 均值方法的模糊聚类
genfis1	基于网格分割方法的模糊推理系统建模
genfis2	基于减聚类方法的模糊推理系统建模
subclust	数据的模糊减聚类

1．anfis

功能：利用 Sugeno 型模糊推理系统建模。

语法：

```
[fismat,error1,stepsize] = anfis(trnData)
[fismat,error1,stepsize] = anfis(trnData,fismat)
[fismat1,error1,stepsize] = anfis(trnData,fismat,trnOpt,dispOpt)
[fismat1,error1,stepsize,fismat2,error2] = anfis(trnData,trnOpt,dispOpt,chkData)
[fismat1,error1,stepsize,fismat2,error2] = anfis(trnData,trnOpt,dispOpt,chkData, optMethod)
```

说明：当未指定验证数据时，函数输出为 3 维向量；指定验证数据时，输出为 5 维向量。

输入参数说明：

trnData：训练学习的输入输出数据矩阵，最后一列代表输出数据，其余每一列代表一个输入数据。

fismat：指定初始的模糊推理系统的名称，如果无此参数，则系统缺省使用 genfis1 作为初始的模糊推理系统。

trnOpt：训练学习的选项，它是一个 5 维向量，每个元素意义如下：

- trnOpt(1)：训练次数，缺省为 10；
- trnOpt(2)：期望误差，缺省为 0；
- trnOpt(3)：初始步长，缺省为 0.01；
- trnOpt(4)：步长递减速率，缺省为 0.9；
- trnOpt(5)：步长递减速率，缺省为 1.1。

如果 trnOpt 的任一元素为 NaN（非数值），则训练使用缺省的参数。学习训练过程在训练参数得到指定值或训练误差得到期望误差时停止。训练过程的步长控制使用如下策略：如果误差连续四次减小，则增加步长；如果误差变化为连续两次出现震荡，即减少步长。

dispOpt：指定训练执行过程中 MATLAB 命令窗口显示选项，它为 4 维向量，每个元素意义如下：

- dispOpt(1)：显示 ANFIS 的信息，缺省为 1；
- dispOpt(2)：显示误差测量，缺省为 1；

- dispOpt(3)：显示训练步长，缺省为 1；
- dispOpt(4)：显示最终结果，缺省为 1。

当某个元素为 0 时就不显示该内容。如果为 1 或 NaN 或省略，则显示。

chkData：提供检验的数据。

输出参数说明：

fismat1：学习完成后得到的对应最小均方根误差的模糊推理系统；

error1：训练的均方根误差向量；

stepsize：训练步长向量；

fismat2：当提供检验数据时，函数返回的对检验数据具有最小均方根误差的模糊推理系统；

error2：检验数据对应的最小均方根误差向量。

【例 1】 ANFIS 输出数据与训练数据比较，如图 6-31 所示。

```
x = (0:0.1:10)';
y = sin(2*x)./exp(x/5);
trnData = [x y];
numMFs = 5;
mfType = 'gbellmf';
epoch_n = 20;
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,20);
plot(x,y,x,evalfis(x,out_fismat));
legend('Training Data','ANFIS Output');
```

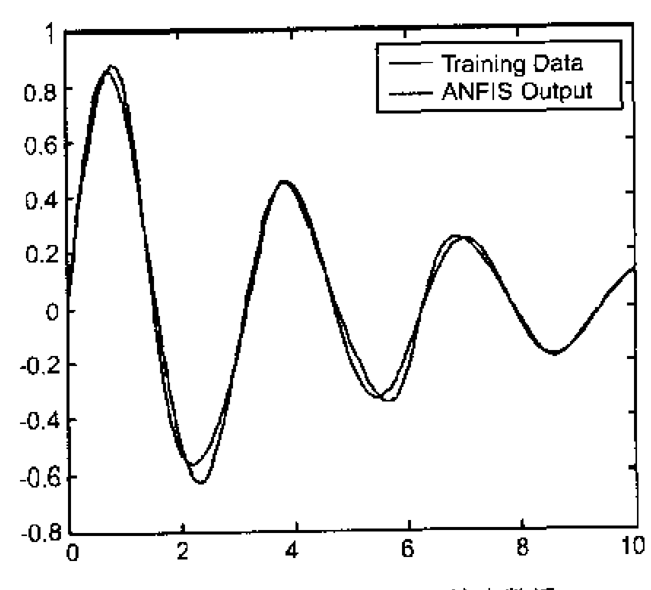


图 6-31 训练数据与 ANFIS 输出数据

2. fcm

功能：利用模糊 C 均值方法的模糊聚类。

语法：[center,U,obj_fcn] = fcm(data,cluster_n)



```
[center,U,obj_fcn] = fcm(data,cluster_n,options)
```

说明：输入参数说明：

data：给定的数据集；

cluster_n：聚类中心的个数；

输出参数说明：

center：迭代后得到的聚类中心；

U：所有数据点对聚类中心的隶属度矩阵；

obj_fcn：目标函数值在迭代过程中的变化值。

可选参数 options 为四维向量，包含若干参数，定义如下：

- options(1) ：分割矩阵的指数，缺省为 2；
- options(2) ：最大迭代数，缺省为 100；
- options(3) ：迭代停止的误差控制准则，缺省 1e-5
- options(4) ：迭代过程中的信息显示，缺省为 1，即显示。

【例 2】 将一组随机数据利用模糊 C-均值聚类方法分为两类，分类结果如图 6-32 所示。

```
data = rand(100, 2);
[center,U,obj_fcn] = fcm(data, 2);
plot(data(:,1), data(:,2),'o');
maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2, :) == maxU);
line(data(index1,1), data(index1, 2), 'linestyle', 'none',
'marker', '*', 'color', 'r');
line(data(index2,1), data(index2, 2), 'linestyle', 'none',
'marker', 'o', 'color', 'r');
```

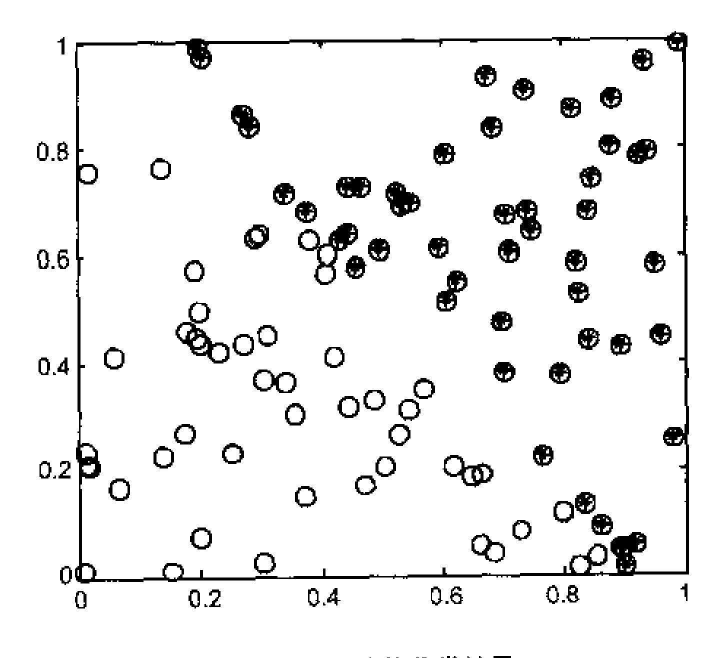


图 6-32　C-均值分类结果

3. genfis1

功能：基于网格分割方法的模糊推理系统建模。

语法：fismat = genfis1(data)

fismat = genfis1(data,numMFs,inmftype, outmftype)

说明：genfis1 采用网格分割的方式根据给定的数据生成模糊推理系统，可以与 anfis 函数合用。有 genfis1 生成的模糊推理系统的输入输出隶属度函数曲线都在保证覆盖整个输入输出的空间基础上进行均匀分割，其输入输出隶属度函数的类型可以在使用时指定。输入参数说明：

data：给定的输入输出数据集合；

numMFs：整数向量用于指定输入输出隶属度函数的个数；

inmftype：输入隶属度函数的类型；

outmftype：输出隶属度函数的类型。

如果只使用一个参数，则使用缺省的，即隶属度函数个数为 2，类型为钟型。

【例 3】

```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
numMFs = [3 7];
mfType = str2mat('pimf','trimf');
fismat = genfis1(data,numMFs,mfType);
[x,mf] = plotmf(fismat,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('input 2 (trimf)');
```

输出的隶属度函数曲线如图 6-33 所示。

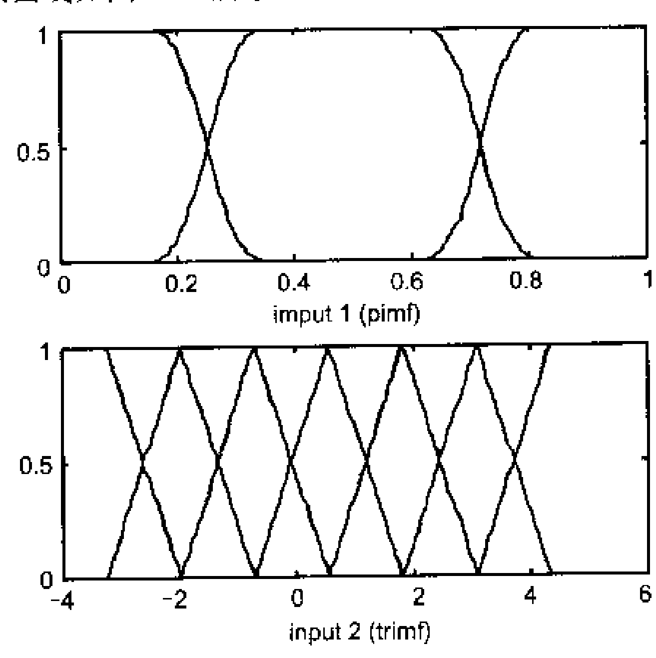


图 6-33　隶属度函数曲线

4．genfis2

功能：基于减聚类方法的模糊推理系统建模。

语法：fismat = genfis2(Xin,Xout,radii)

fismat = genfis2(Xin,Xout,radii,xBounds)

fismat = genfis2(Xin,Xout,radii,xBounds,options)

说明：输入参数说明：

Xin：输入数据集；

Xout：输出数据集；

radii：用于假定数据点位于一个单位超立方体内的条件下，指定数据向量的每一维聚类中心影响的范围。每一维取值在 0 到 1 之间。

Xbounds：为 2*N 维的矩阵，其中 N 为数据的维数。

options：参数向量，说明如下：

- options(1) = quashFactor：quashFactor 用于与聚类中心的影响范围 radii 相乘，用以决定某一聚类中心邻近的那些数据点被排除作为聚类中心的可能性，缺省为 1.25；
- options(2) = acceptRatio：acceptRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性值高于第一个聚类中心可能性值的一定比例，只有高于这个比例才能被作为新的聚类中心。缺省为 0.5；
- options(3) = rejectRatio：rejectRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性值低于第一个聚类中心可能性值的一定比例，只有低于这个比才能被排除作为新的聚类中心。缺省为 0.15；
- options(4) = verbose：如果 verbose 为非零值，则聚类过程的有关信息将显示出来，否则将不显示。

【例 4】 函数的几种调用方法：

```
fismat = genfis2(Xin,Xout,0.5)
fismat = genfis2(Xin,Xout,[0.5 0.25 0.3])
fismat = genfis2(Xin,Xout,0.5,[-10 -5 0; 10 5 20])
```

5．subclust

功能：数据的模糊减聚类。

语法：[C,S] = subclust(X,radii,xBounds,options)

说明：输入参数说明：

X：包含用于聚类的数据，X 的每一行为一个数据向量；

radii：用于假定数据点位于一个单位超立方体内的条件下，指定数据向量的每一维聚类中心影响的范围。每一维取值在 0 到 1 之间。

Xbounds：为 2*N 维的矩阵，其中 N 为数据的维数。

options：参数向量，说明如下：

- options(1) = quashFactor：quashFactor 用于与聚类中心的影响范围 radii 相乘，用以决定某一聚类中心邻近的那些数据点被排除作为聚类中心的可能性，缺省为 1.25。
- options(2) = acceptRatio：acceptRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性值高于第一个聚类中心可能性值的一定比例，只有高于这个比才

能被作为新的聚类中心。缺省为 0.5。

- options(3) = rejectRatio：rejectRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性值低于第一个聚类中心可能性值的一定比例，只有低于这个比才能被排除作为新的聚类中心。缺省为 0.15。
- options(4) = verbose：如果 verbose 为非零值，则聚类过程的有关信息将显示出来，否则将不显示。

返回参数 C 为聚类中心向量，向量 S 包含了数据点每一维聚类中心的影响范围。

【例 5】

```
[C,S] = subclust(X,0.5)
[C,S] = subclust(X,[0.5 0.25 0.3],[],[2.0 0.8 0.7])
```

6.5　模糊逻辑工具箱接口及示例函数

MATLAB 的模糊逻辑工具箱提供了与 Simulink 的无缝连接，在模糊逻辑工具箱中建立了模糊推理系统后，可以立即在 Simulink 中对其进行仿真。在 Simulink 中有相应的模糊逻辑控制方块，将该方块拷贝到用户建立的 Simulink 仿真模块中，即可实现模糊逻辑工具箱提供的与 Simulink 的无缝连接。在模糊逻辑工具箱图形用户界面示例中提供了丰富的 Simulink 与模糊逻辑控制工具箱的连接示例。

示例中提供的一个水箱水位控制系统的 Simulink 仿真模块，其系统框图如图 6-34 所示。

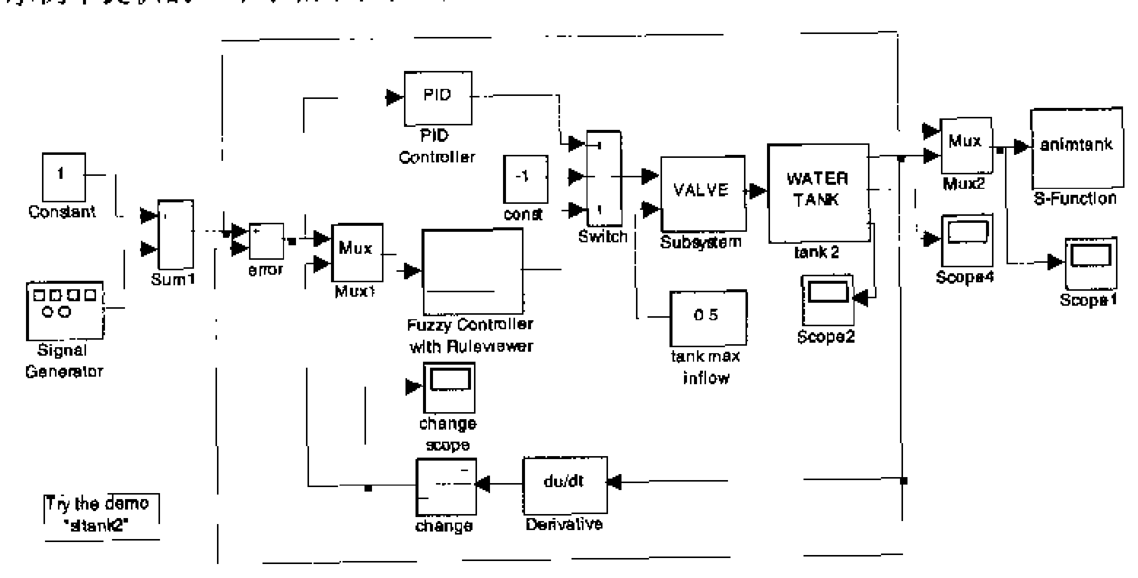


图 6-34　水箱水位控制系统框图

MATLAB 提供了与 C 语言的接口。MATLAB 通过两个 C 语言文件实现了 C 语言环境下对模糊逻辑工具箱系统功能的调用，这两个文件为：fismain.c 和 fis.c。通过对这两个函数的调用，可以直接读取磁盘中的.fis 文件，并读取输入数据文件进行模糊推理。在 UNIX 系统下编译命令为

```
%    cc -O -o fismain fismain.c –lm
```




（%为 UNIX 系统提示符，或为 DOS 或 WINDOWS 的命令窗口的提示符）

编译以后可以输入 fismain 看看它是如何工作的：

% fismain

UNIX 返回：

% Usage: fismain data_file fis_file

这表明 fismain 需要两个输入文件来工作。

例如：假设一个模糊推理系统在系统中文件名为：mam21.fis。可以使用 MATLAB 来准备输入数据如下：

```
[x, y] = meshgrid(-5:5, -5:5);
input_data = [x(:) y(:)];
save fis_in input_data –ascii
```

该程序将输入数据存为 ASCII 文件 fis_in，接着可以使用 fismain 调用这些文件：

% fismain fis_in mam21.fis

这行命令可以在屏幕上产生 121 行输出数据，为了方便观察，可以将这些数据输入到一个文件中：

% fismain fis_in mam21.fis > fis_out

可以将 fismain 的模糊推理结果与 MATLAB 中的 evalfis 函数的结果比较：

```
fismat = readfis('mam21');
matlab_out = evalfis(input_data, fismat);
load fis_out
max(max(matlab_out - fis_out))
```

MATLAB 返回为：

```
ans =
    4.9583e-13
```

从结果可以看出利用 fismain 的结果与 evalfis 函数的推理结果基本相同。

有关模糊逻辑工具箱与 C 语言的接口的注意事项如下：

- 模糊逻辑工具箱与 C 语言的接口函数遵循 ANSI C 标准，只要求 C 编译器中定义 _STDC_即可。
- 接口函数只支持 MATLAB 模糊逻辑工具箱提供的隶属度函数类型和模糊推理运算。要扩展与 C 语言的接口，用户可以自己改变 fis.c 文件。通用接口函数见表 6-5。

表 6-5　接口函数及示例

函数名	说　明
Fuzblock	Simulink 模糊逻辑库
Sffis	Simulink 模糊推理 S 型函数
Fuzdemos	模糊逻辑工具箱图形用户界面示例

1．fuzblock

该命令打开一个 Simulink 模糊逻辑库，打开窗口如图 6-35 所示。

双击左边两幅图可以打开模糊逻辑控制对话框，如图 6-36 和图 6-37 所示。

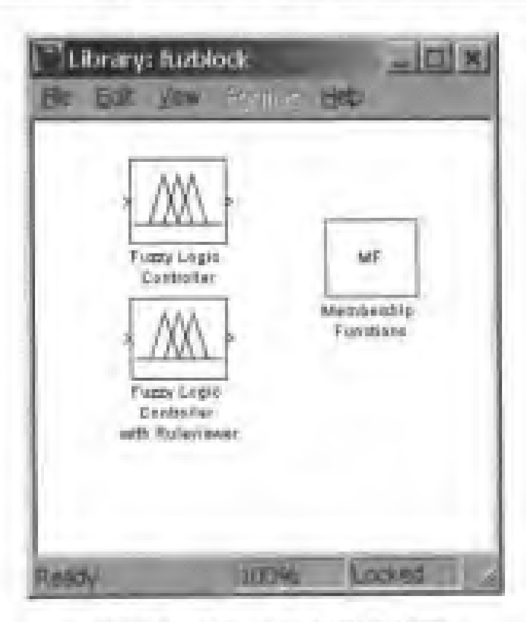


图 6-35 Simulink 模糊逻辑库



图 6-36 模糊逻辑控制对话框

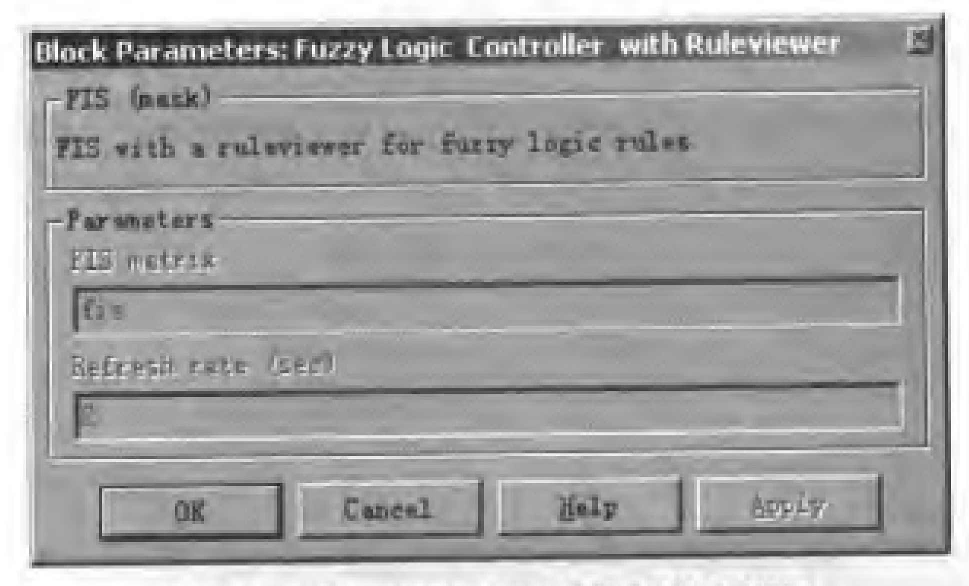


图 6-37 具有模糊规则观察器的模糊逻辑控制对话框



双击右边图可以打开 Simulink 模糊逻辑库隶属度函数观察窗口，如图 6-38 所示。

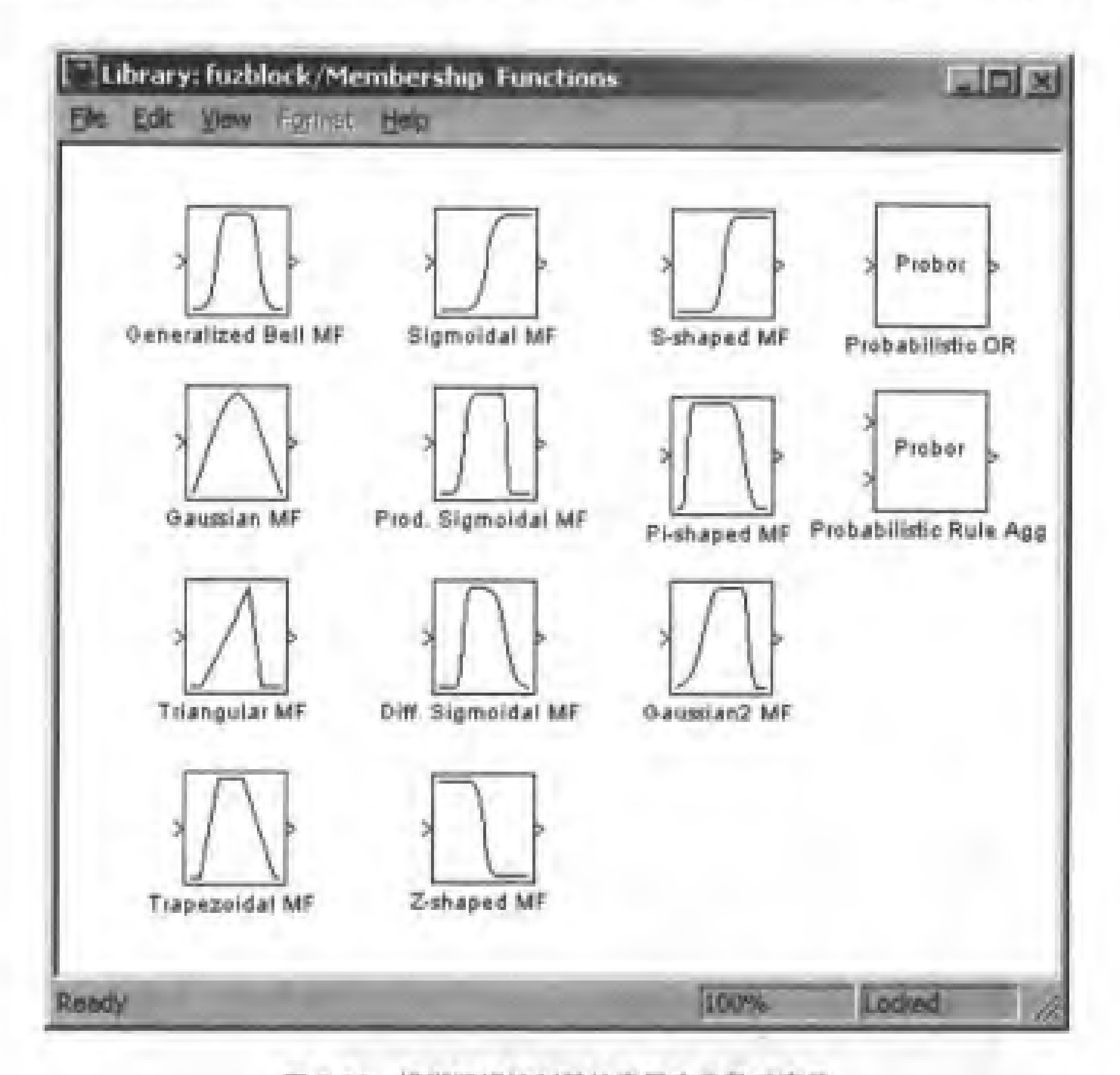


图 6-38　模糊逻辑控制器的隶属度函数观察器

2．sffis

功能：Simulink 模糊推理 S 型函数。

语法：output = sffis(t,x,u,flag,fismat)

说明：在很多情况下模糊逻辑控制模块为用户的模糊逻辑推理系统生成定制的块图。然而模糊向导不能处理这些定制的模块，在这些情况下模糊逻辑控制模块使用 S 函数 sffis 模拟模糊逻辑推理系统。

参数 t,x,flag 为标准的 Simulink 中 S 函数的参数。u 为 MATLAB 工作空间中标准的 FIS 变量。

3．fuzdemos

功能：模糊逻辑工具箱图形用户界面示例。

通过在命令行中输入 c 可以打开模糊逻辑工具箱的示例窗口，如图 6-39 所示。

选择演示的条目，可以打开各自的演示窗口，例如选择 Membership function gallery 打开的隶属度函数图形窗口如图 6-40 所示。



图 6-39　模糊逻辑工具箱的示例窗口

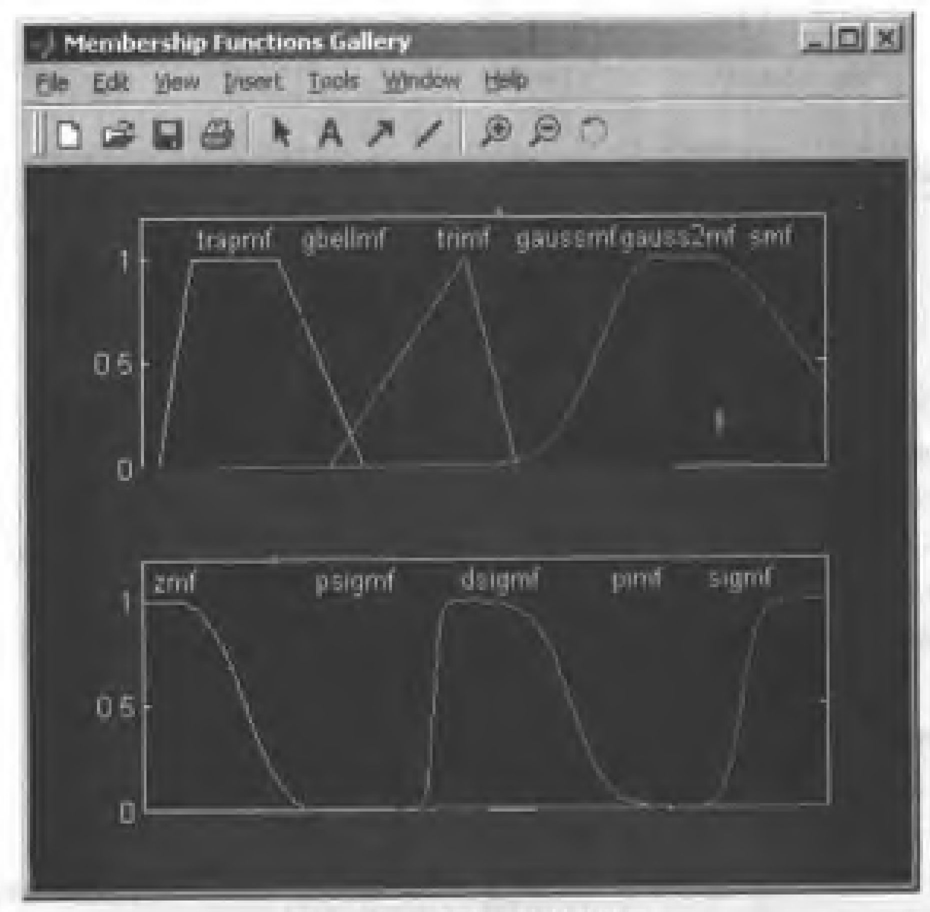


图 6-40　隶属度函数图形演示

第7章 非线性控制设计模块

许多控制系统都具有非线性特性。例如随动系统的齿轮传动具有齿隙和摩擦等，许多执行机构都不可能无限制地增加其输出功率，因此就存在饱和非线性特性。以上所举的例子中的非线性是由于系统的不完善而产生的，这种不完善实际上是不可避免的。有些非线性是系统动态特性本身所固有的。例如高速运动的机械手各关节之间有哥氏力的耦合，这种耦合是非线性的，如果要研究机械手高速运动的控制就必须考虑非线性耦合。电力系统中传输功率与各发电机之间相角差的正弦成正比，如果要研究电力系统中的大范围运动时，就必须考虑非线性特性的影响。还有一类对象本身虽然是线性的，但为了对它进行高质量的控制，常常在控制系统中有意识地引进非线性的控制规律。

为了解决上述的非线性问题，需要针对非线性系统进行控制器优化和仿真。MATLAB 提供了非线性控制设计模块（Nonlinear Control Design Blockset），简称 NCD 模块。为非线性系统进行控制器优化和仿真提供了有效的工具。该模块以 Simulink 的形式集成了基于图形界面的非线性系统控制器优化和仿真功能。

7.1 NCD 模块的使用

7.1.1 建立闭环系统方框图

将 NCD 模块与闭环系统的输出相连，得到图 7-1 所示的 NCD 闭环系统方框图。可以通过在 MATLAB 的命令行中键入：ncdtut1 打开相应的窗口。

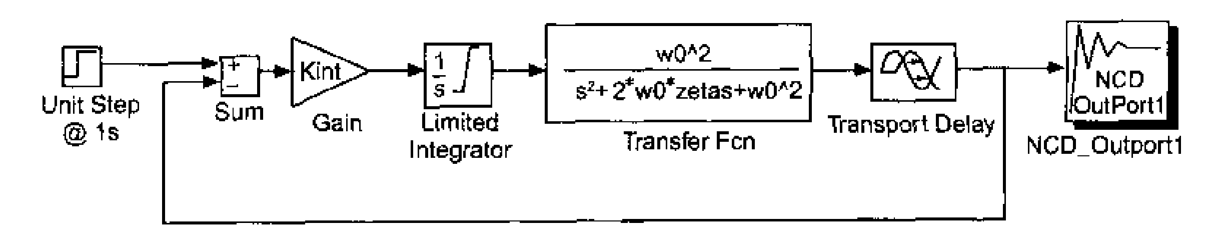


图 7-1 NCD 闭环系统方框图

接下来必须初始化 MATLAB 变量，以供优化仿真使用。在 MATLAB 提示符下输入：

```
zeta = 1;
w0 = 1;
Kint = 0.3;
```

7.1.2 设置约束条件

使用鼠标双击框图中的 NCD 模块，可以打开一个 NCD 模块的时域性能窗口，如图 7-2 所示。

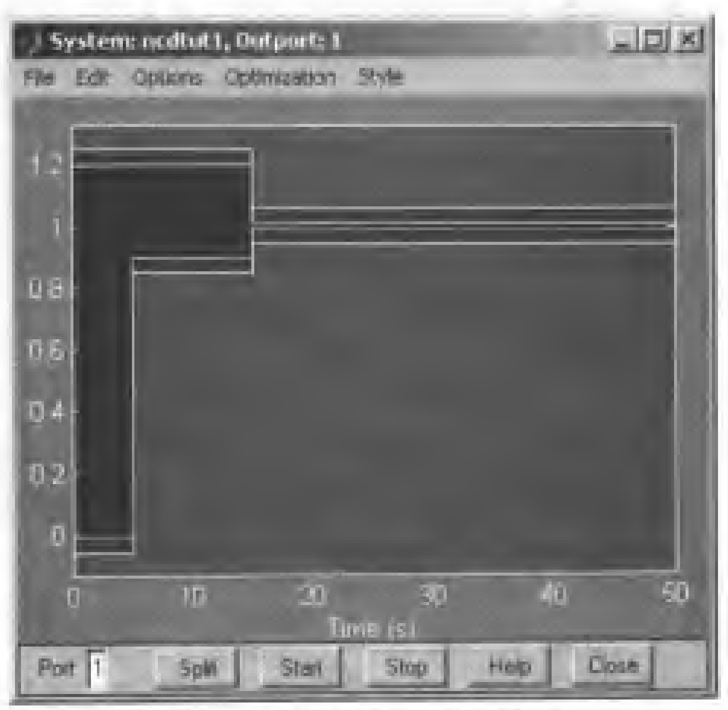


图 7-2 NCD 模块的时域性能约束

该窗口图形显示了变量时域性能约束，横坐标为时间轴，纵坐标为约束变量的取值。水平的长线条用语指定变量约束的上下界，可以使用鼠标改变各线段长度和水平位置。修改后的窗口如图 7-3 所示。

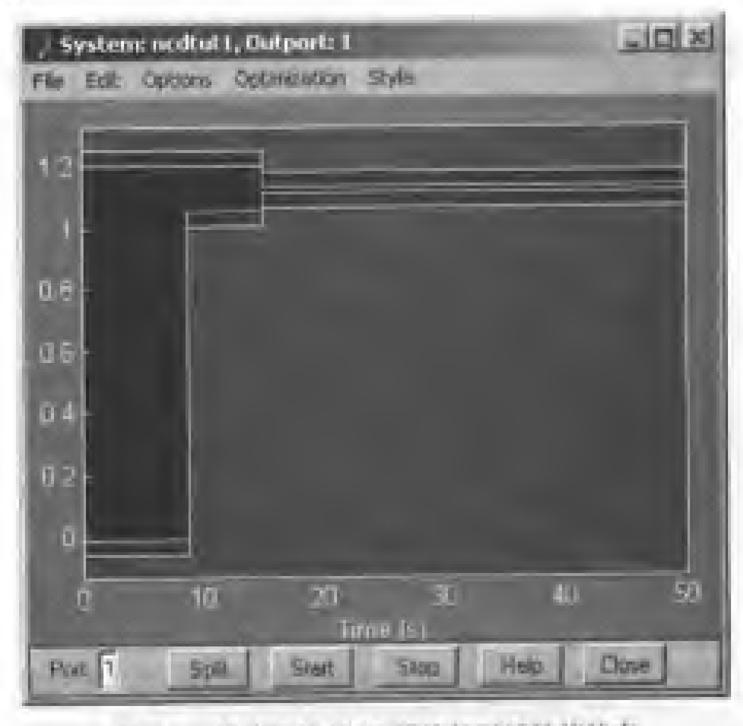


图 7-3 修改后的 NCD 模块的时域性能约束

关于该时域性能约束窗口的菜单说明如下：

1．文件（File）菜单

Load：从文件中加载约束数据；

Save：保存约束数据到磁盘文件中；

Close：关闭当前窗口；

Print：打印约束数据。

2．编辑（Edit）菜单

Undo：撤消最近操作；

Edit Constrains：打开约束数据窗口；

Delete Plots：删除变量响应曲线。

选中任一条约束边界线后，再选择编辑菜单中的 Edit Constrains，可以打开相应的约束数据窗口，如图 7-4 所示。

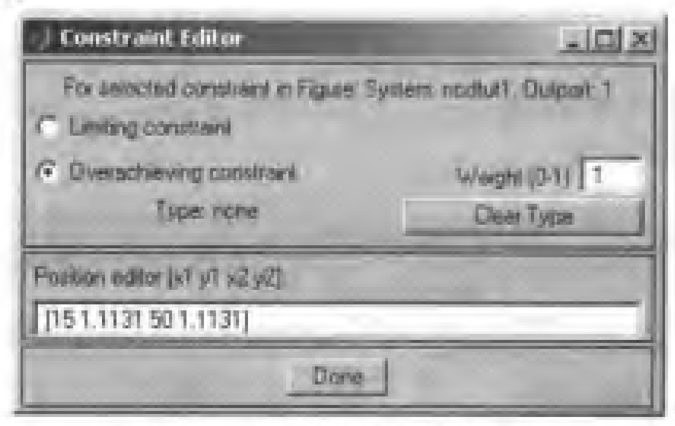


图 7-4　约束数据参数设置窗口

3．选项（Options）菜单

Initial response：设置初始响应；

Reference input：设置参考输入；

Step response：设置阶跃响应特性；

Time range：设置时域长度范围；

Y-Axis：设置 Y 轴变量；

Refresh：清除计算结果。

选择 Time range 后，可以打开如图 7-5 所示的窗口，在该窗口中可以对时间轴的长度范围和标注进行修改。

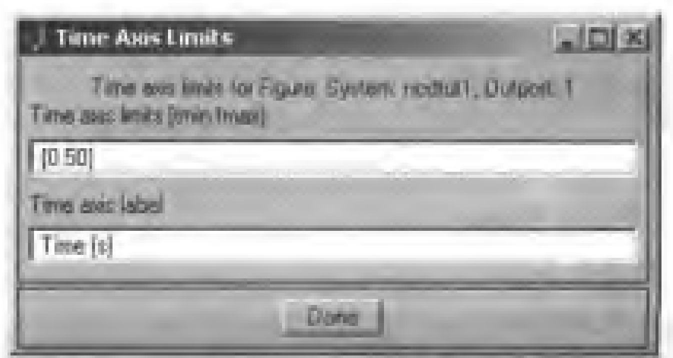


图 7-5　时间轴设置窗口

选择了 Y-Axis 后，可以打开如图 7-6 的窗口，在该窗口中可以对 Y 轴约束变量的显示范围和 Y 轴的标注进行修改。

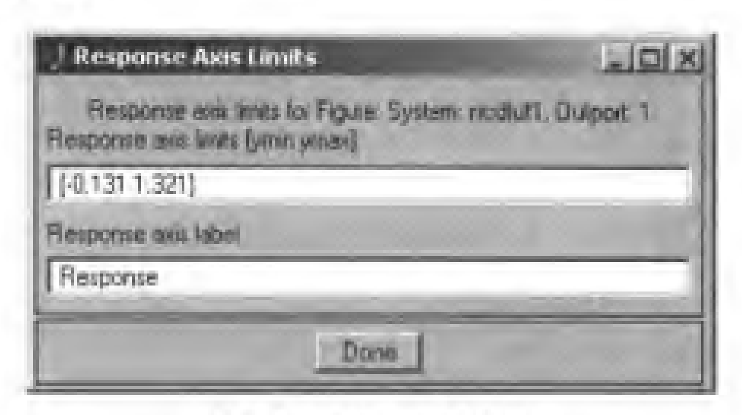


图 7-6　Y 轴变量设置窗口

选择 Step response 菜单，可以进入阶跃响应约束参数设置窗口，如图 7-7 所示。在该窗口中可以对阶跃响应曲线的 Setting Time（调节时间）、Percent over shoot（超调量）、Rise Time（上升时间）等参数进行设置。

Step Response
Input step response characteristics.
Settling time 15　Rise time 5
Percent settling 5　Percent rise 90
Percent overshoot 20　Percent undershoot 1
Step time 0　Final time 50
Initial output 0　Final output 1
Done　Revert　Help

图 7-7　阶跃响应曲线参数设置窗口

4. 优化（Optimization）菜单

Start：开始优化和仿真；

Stop：停止优化和仿真；

Parameters：设置优化参数；

Uncertainty：设置不确定参数范围。

选择 Parameters 菜单，会弹出如图 7-8 所示的参数设置窗口。

其中参数意义如下：

Tunable Variables：优化变量名称；

Lower bounds：优化变量下界；

Upper bounds：优化变量上界；

Discretization interval：离散化区间长度；

Variable Tolerance：优化变量误差容许限度；

Constraint Tolerance：约束误差容许限度。

Optimization Parameters

Tunable Variables:

Lower bounds (optional):

Upper bounds (optional):

Discretization interval: 1

Variable Tolerance: 0.001

Constraint Tolerance: 0.001

Display optimization information

Stop optimization as soon as the constraints are achieved

Compute gradients with better accuracy (slower)

Done

Revert

Help

Note: Simulation parameters are entered in the SIMULINK system.

图 7-8　优化参数设置窗口

5. 类型（Style）菜单

该菜单用于设置作图的参数。

7.1.3　开始优化计算

在完成初始化参数设置和优化参数设置后，可以开始进行优化计算。选择优化菜单的 Start，开始优化计算。结果系统响应曲线如图 7-9 所示。可以看出，系统的响应曲线逐渐满足约束的要求。

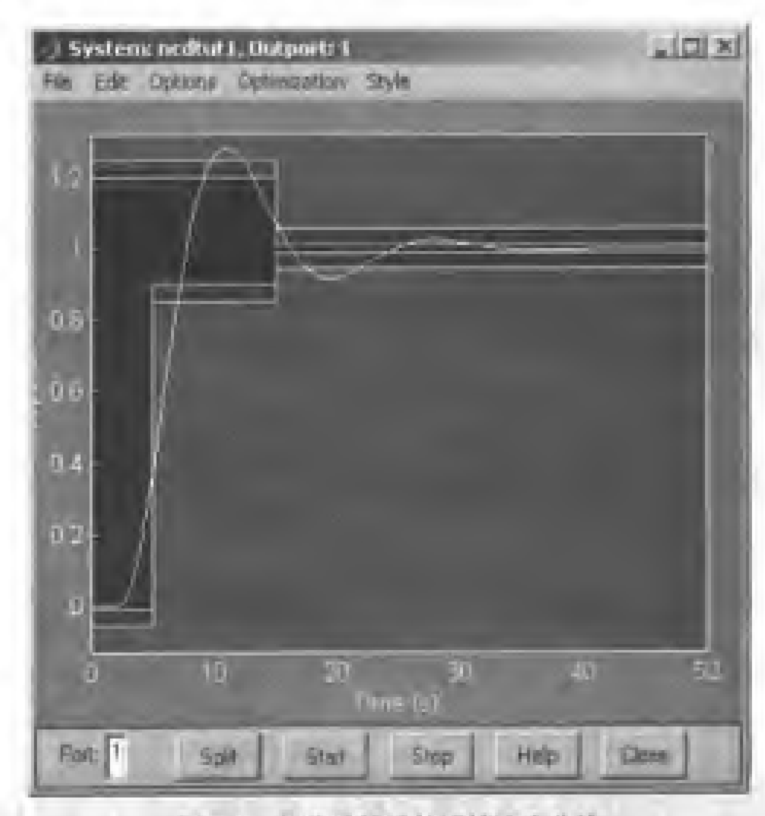


图 7-9　优化计算后的系统响应曲线

7.2 NCD 模块应用实例

7.2.1 问题提出

使用 NCD 模块解决一个闭环系统辨识问题。具体的讲，就是估计一个倒摆的质量和长度。该系统具有一个圆柱形的金属长竿，下端连接到一个由马达驱动的小车上，小车在一条直线轨道上行驶，如图 7-10 所示。长竿的初始质量为 0.21kg，长度为 0.61m，并且系统在 LQR 控制下是稳定的。

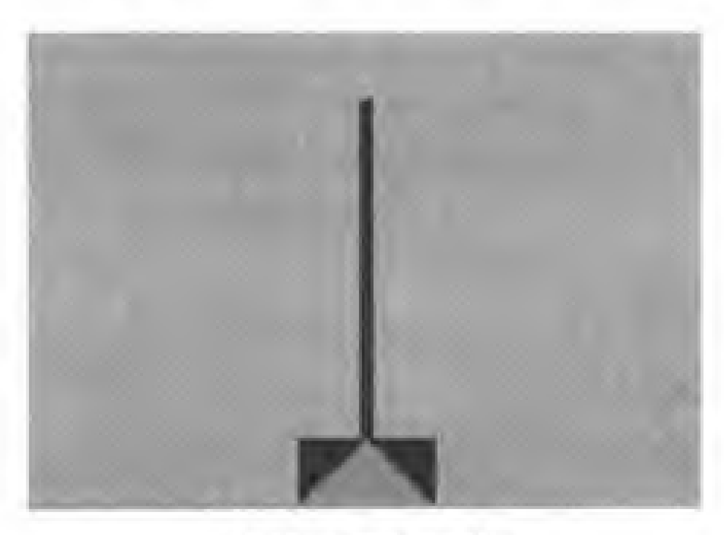


图 7-10　倒摆系统示意图

7.2.2 NCD 模块启动

在 MATLAB 提示符下输入 ncdtut2，得到系统闭环方框图，如图 7-11 所示。

在图 7-11 中，[T,ThetaHat]和[T,YHat]为倒摆系统的实际测量数据，两个 NCD 模块 CD_Outport1 和 NCD_Outport2 分别用于对倒摆角度误差和摆杆误差进行约束，对应的优化参数为倒摆质量 m 和长度 l。反馈增益矩阵 Klqr 为基于线性模型设计的 LQR 调节器矩阵。

接着，必须定义系统的初始化参数，和载入系统可观数据。在 MATLAB 命令行中输入如下命令：

```
penddata
l = 0.61/2;   %摆长重心的高度，即摆长的一半
m = 0.21;     %质量单位为千克
```

7.2.3 设置约束条件

双击 NCD 模块的两个输出框图，显示小车位置和倒摆角度的时域性能约束如图 7-12 和图 7-13 所示。

修改系统约束条件。在约束下界的线段上单击鼠标右键，弹出修改窗口，输入[0 –1 5 –1]作为新的下界；在约束上界的线段上单击鼠标右键，弹出修改窗口，输入[0 1 5 1]作为新的上

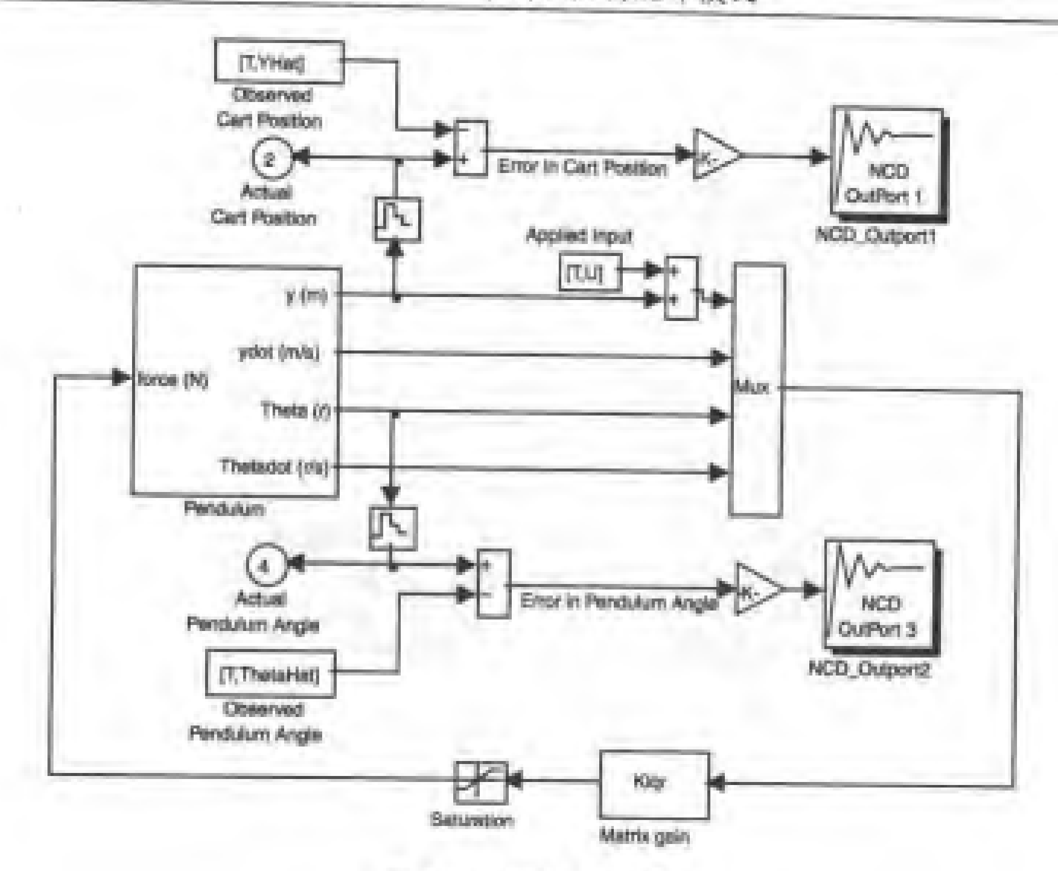


图 7-11 闭环系统方框图



图 7-12 小车位置时域性能约束

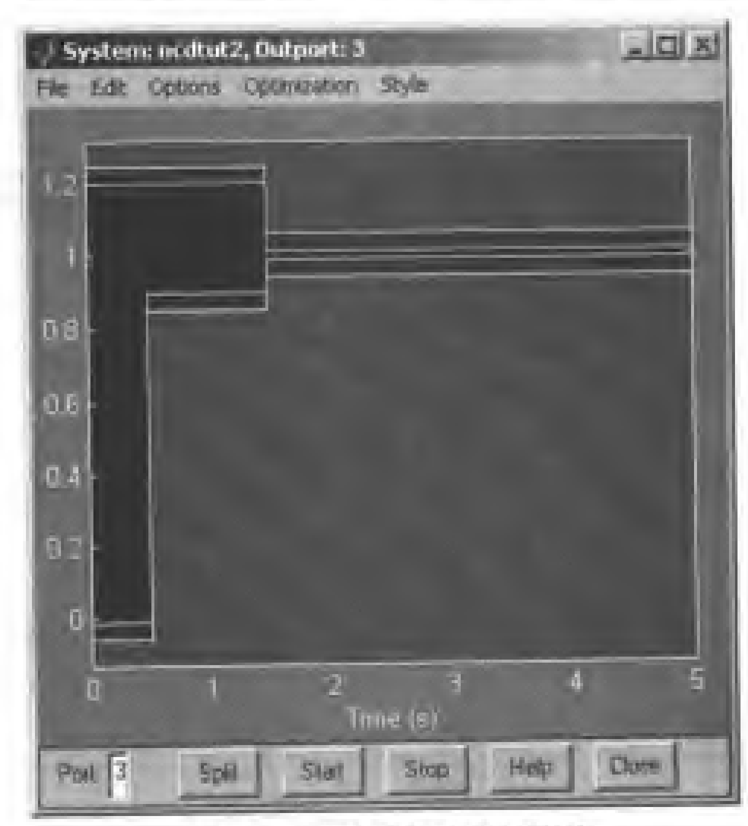


图 7-13 倒摆角度的时域性能约束

界。由于现在下界在 Y 轴缺省的范围之外，所以需要修改 Y 轴显示范围，在菜单栏上选择 Options 下的 Y axis...，弹出的修改窗口中输入新的 Y 轴范围：[-1.5 1.5]。新的时域性能约束图形如图 7-14 所示。

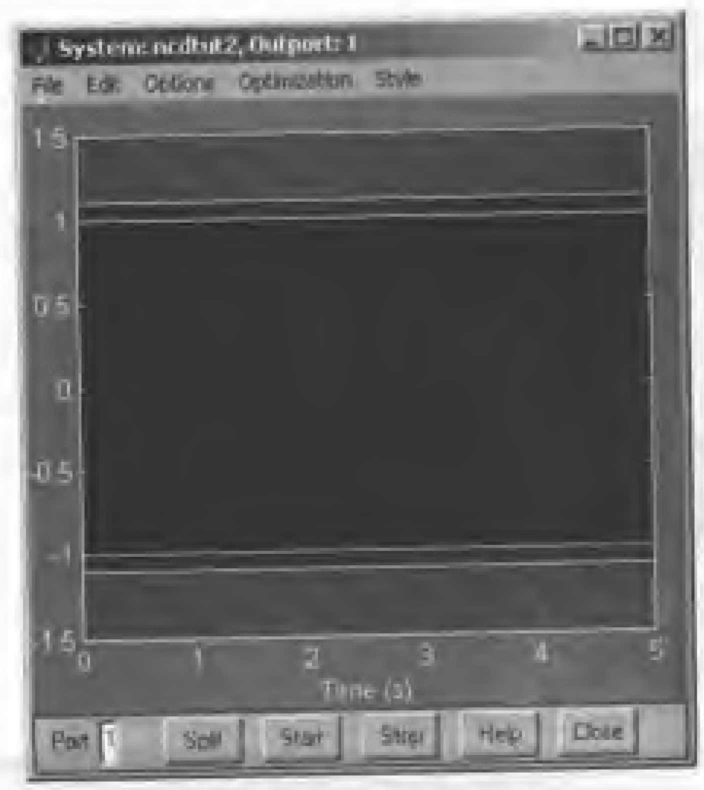


图 7-14 修改了的时域性能约束

当双击 NCD 模块时，系统自动生成变量 ncdStruct，约束的下限储存在 ncdStruct.CnstrLB 中，上限储存在 ncdStruct.CnstrUB 中，Y 轴显示范围储存在 ncdStruct.RngLmts 中。在约束编辑对话框和 Y 轴范围对话框中修改的数据储存在该结构体中。可以通过在 MATLAB 的命令行中输入 ncdStruct.CnstrLB 直接观察下界的储存方式。因为 ncdStruct 直接储存在 MATLAB 的工作空间中，因此我们强烈建议用户直接在 MATLAB 的工作空间中修改这些数据。

接下来需要对优化参数进行设置，打开优化参数对话框，输入参数。修改后的对话框应该如图 7-15 所示。

图 7-15　修改过后的优化参数对话框

7.2.4　优化计算

在设置完优化变量和调整了上下限后，可以开始优化计算了。在运行 WINDOWS 2000 的 PC 机上进行优化计算 MATLAB 输出如下：

```
Processing uncertainty information.
Uncertainty turned off.
Setting up call to optimization routine.
Start time: 0 Stop time: 5.
There are 404 constraints to be met in each simulation.
There are 2 tunable variables.
There are 1 simulations per cost function call.
Creating simulink model NCDmodel for gradients...Done
f-COUNT MAX{g} STEP Procedures
5 0.466256 1
10 -0.421419 1 Hessian modified
15 -0.392145 1
20 -0.477709 1
25 -0.478421 1 Hessian modified twice
30 -0.473284 1 Hessian modified twice
```

35 -0.473988 1 Hessian modified twice
44 -0.473985 0.0625 Hessian modified twice
51 -0.474014 0.25 Hessian modified
60 -0.474371 0.0625 Hessian modified twice
93 -0.474371 -3.73e-009 Hessian modified twice
126 -0.474371 -3.73e-009 Hessian modified twice
159 -0.474371 -3.73e-009 Hessian modified twice
164 -0.473248 1 Hessian modified twice
165 -0.477089 1 Hessian modified twice
Optimization Converged Successfully
Active Constraints:
54

因为运行平台的不同，结果可能有些微小的差异。图 7-16 和图 7-17 显示了初始时和结束时的小车位置和倒摆角度误差的响应曲线。

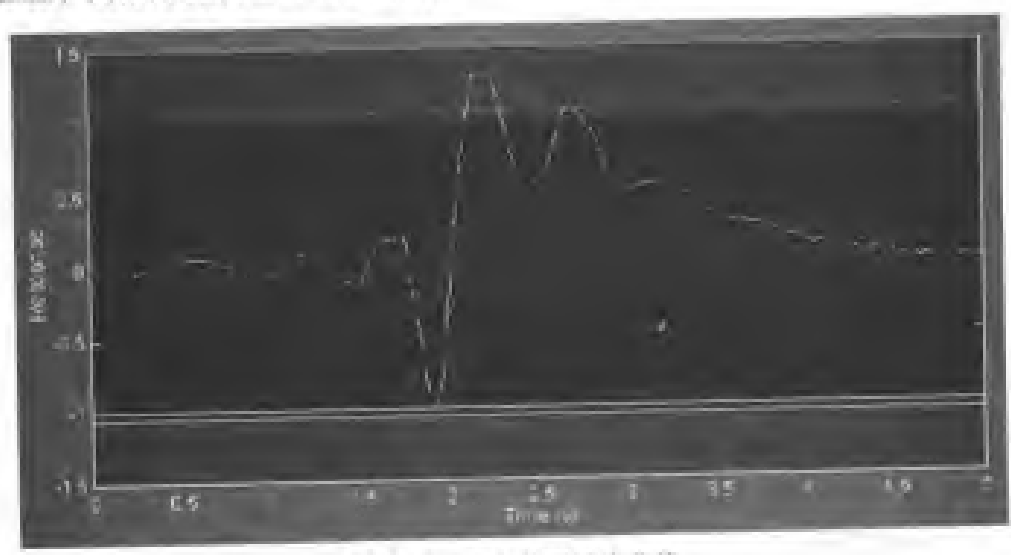


图 7-16　小车位置响应曲线



图 7-17　倒摆角度响应曲线

如果需要了解噪声对优化起什么影响，可以使用如下代码：

(1) 重新初始化：

m = 0.21;

l = 0.61/2;

(2) 在系统中加入随机噪声：

yHat = yHat + 0.001*rand(size(yHat));

ThetaHat = Thetahat + 0.001*rand(size(ThetaHat));

(3) 对系统重新进行优化计算，观察结果的改变。

7.3　NCD 模块几个示例

为了加深对 NCD 模块使用方法的理解，我们再举几个示例程序。通过这几个示例可以加深对 NCD 模块使用的理解。这些示例位于 Demo 中的 NCD 模块的示例中，如图 7-18 所示。

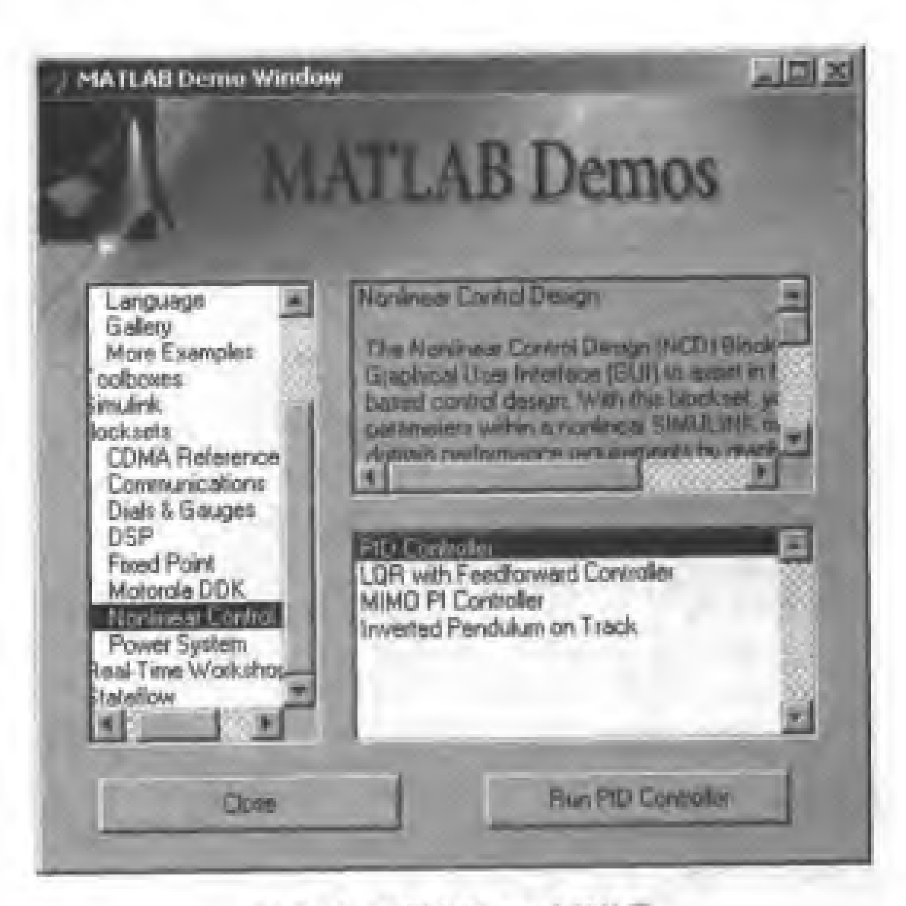


图 7-18　示例在 Demo 中的位置

7.3.1　PID 控制器优化设计示例

首先选择一个 PID 控制器优化设计的问题。通过单击按钮 Run PID Controller 可以运行该示例。打开的 PID 控制系统方框图如 7-19 所示。

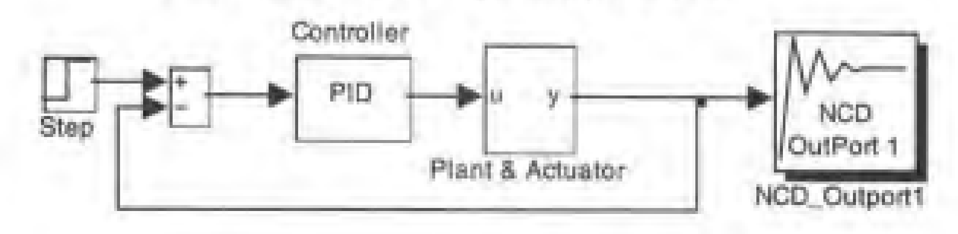


图 7-19 PID 控制系统方框图

双击 ncd1init 方块，对系统进行优化参数的初始化。对系统的输出阶跃响应进行约束，其中时域约束如图 7-20 所示。

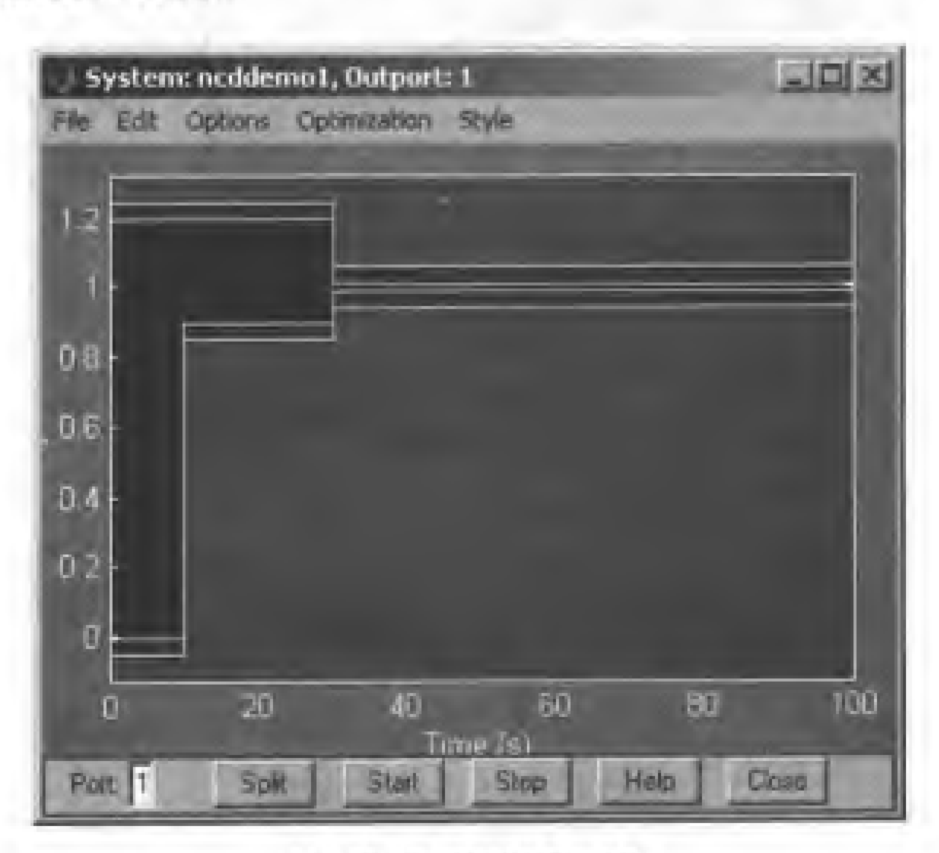


图 7-20 系统输出的时域约束

初始化后，系统设置的优化参数为 Kp、Ki、Kd。参数设置窗口如图 7-21 所示。

为了适应对象的不确定性，示例在初始化后设定了不确定变量，如图 7-22 所示。



图 7-21 优化参数设置窗口

图 7-22 不确定变量设置窗口

也可以不运行 ncd1init，只需自己手工打开相应的参数设置窗口设置相应的参数，或者直接在 MALTAB 中输入相应参数即可。

在完成了以上的参数和变量的设置以后，可以进行 NCD 模块优化和仿真了，选择 Optimization 菜单中的 Start 开始优化计算。

优化在 MATLAB 中返回：

Processing uncertainty information.

Uncertainty turned off.

Setting up call to optimization routine.

Done plotting the initial response.

Start time: 0   Stop time: 100.

There are 205 constraints to be met in each simulation.

There are 3 tunable variables.

There are 1 simulations per cost function call.

f-COUNT	MAX{g}	STEP	Procedures
5	0.368535	1	
10	0.34456	1	Hessian modified twice
15	0.204199	1	
20	0.0846013	1	Hessian modified
25	0.0199867	1	Hessian modified
30	−0.00460567	1	Hessian modified
35	−0.0069396	1	Hessian modified
40	−0.0078251	1	Hessian modified
45	−0.00796174	1	Hessian modified twice
50	−0.00813777	1	Hessian modified twice
58	−0.00816259	0.125	Hessian modified
63	−0.00815719	1	Hessian modified twice
68	−0.00848506	1	Hessian modified twice
69	−0.00850071	1	Hessian modified twice

Optimization Converged Successfully

Active Constraints:

16

32

44

138

相应的优化参数为：

Kp =

1.3365

Ki =

0.1548

Kd =

8.3317

优化后的响应曲线（图中深色的曲线）如图 7-23 所示。

7.3.2 多变量状态反馈系统控制优化

选择 NCD 模块的第二个示例：LQR with Freeforward Controller。打开系统方框图，由于本书版面较小，故对控制方块的位置稍作了调整，其结果如图 7-24 所示。

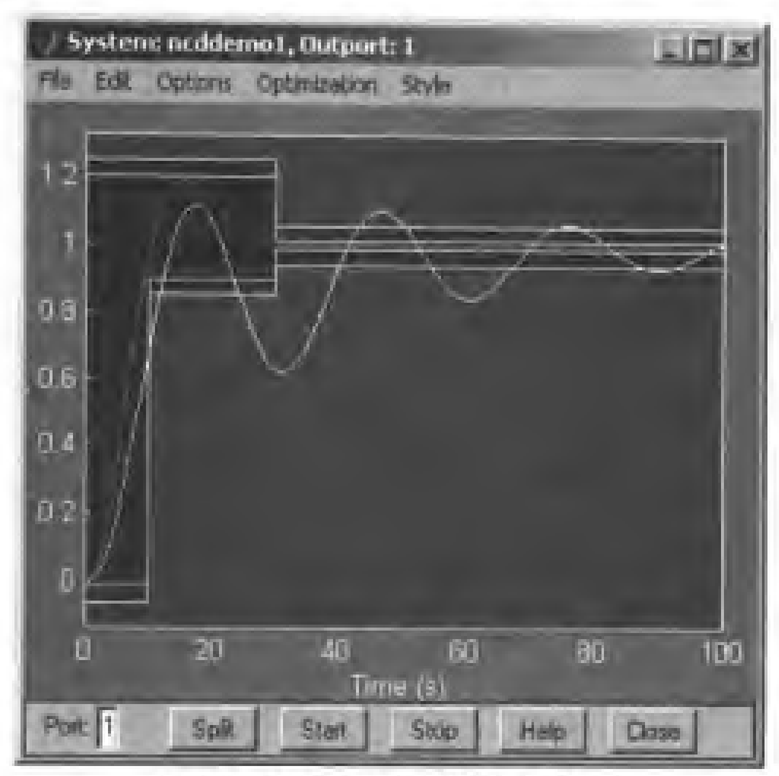


图 7-23　优化前后的系统输出响应曲线

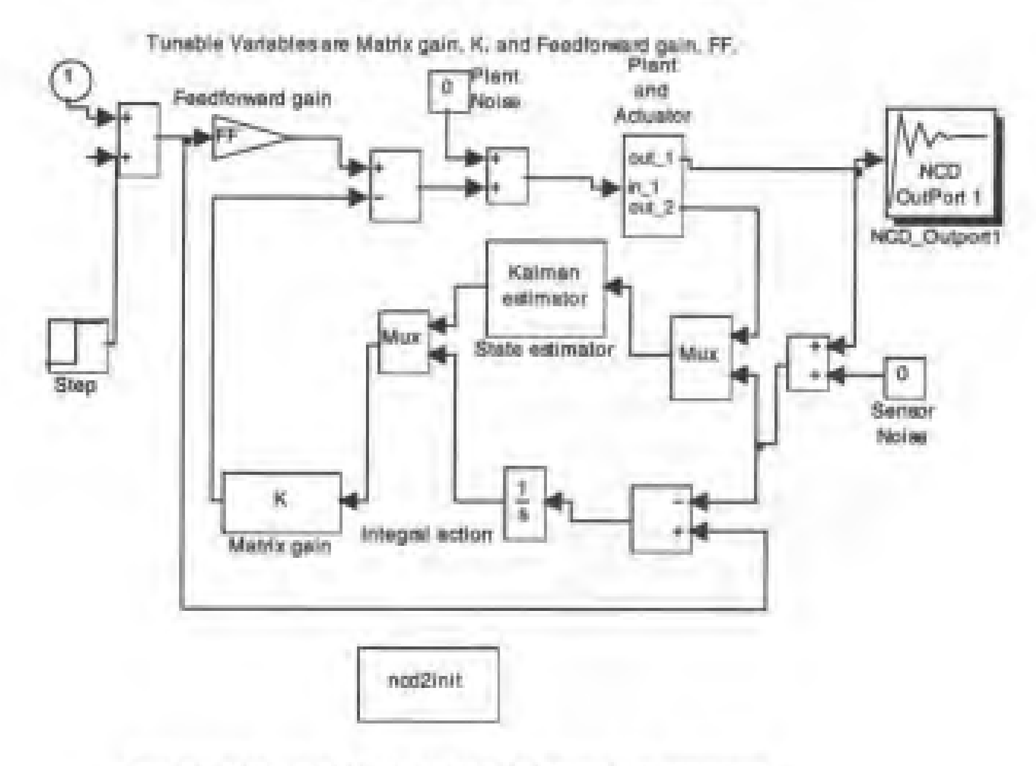


图 7-24　多变量状态反馈系统方框图

对系统进行初始化，双击按钮 ncd2init。初始化后的系统优化参数和不确定变量窗口如图 7-25 和图 7-26 所示。

Optimization Parameters

Tunable Variables: K FF

Lower bounds:

Upper bounds:

Discretization interval: 0.1

Variable Tolerance: 0.01

Constraint Tolerance: 0.001

Display optimization information

Stop optimization as soon as the constraints are achie

Compute gradients with better accuracy (slower)

Done Revert Help

Note: Simulation parameters are entered in the SIMULINK system.

图 7-25　优化参数设置窗口

Uncertain Variables

Uncertain variables: a

Lower bounds: a/2

Upper bounds: 2*a

Number of Monte Carlo simulations: 0

Constrain nominal simulation

Constrain lower bound simulation

Constrain upper bound simulation

Constrain Monte Carlo simulations

Total simulations per step = 1

Done Revert Help

图 7-26　不确定变量设置窗口

NCD 模块中输出 1 的时域性能约束窗口如图 7-27 所示。

接下来可以进行优化计算，结果在 MATLAB 中输出为：

Processing uncertainty information.

Uncertainty turned off.

Setting up call to optimization routine.



```
Done plotting the initial response.
Start time: 0    Stop time: 10.
There are 205 constraints to be met in each simulation.
There are 6 tunable variables.
There are 1 simulations per cost function call.
f-COUNT      MAX{g}         STEP  Procedures
    8        0.615677          1
   16        0.161844          1   Hessian modified twice
   24        0.00139343        1   Hessian modified
   25       -0.00860307        1   Hessian modified
Optimization Converged Successfully
Active Constraints:
   102
   145
```

结果优化参数为：

```
K =
   -1.0585    0.9378    0.0188   -0.2413    0.0786
FF =
    1.2729
```

注：不同的系统上运行结果可能会有一些差异。

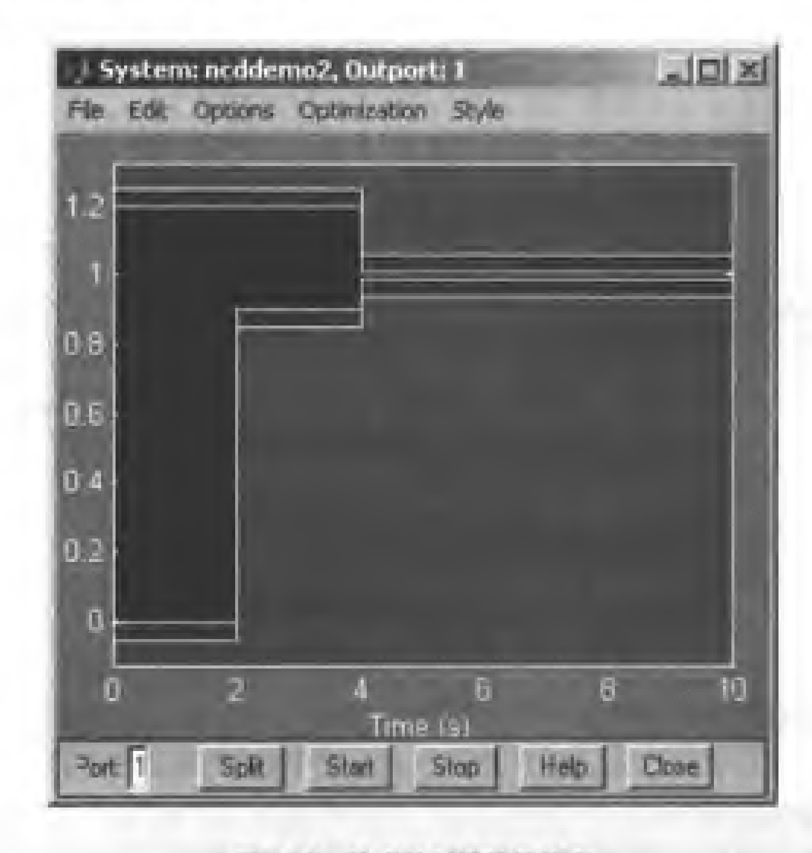


图 7-27 输出的时域性能约束

采用这组优化参数得到的系统阶跃响应曲线如图 7-28 所示。

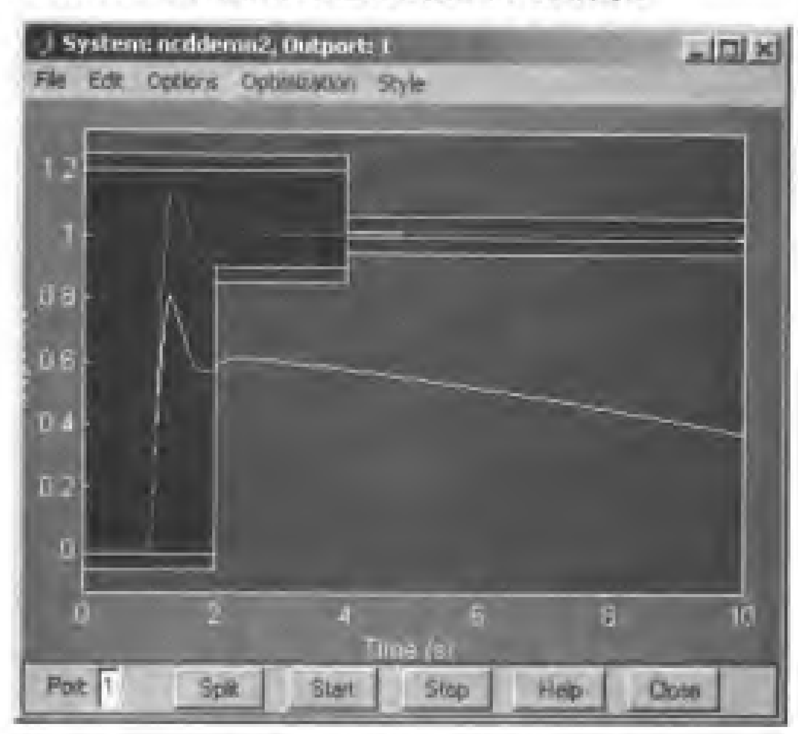


图 7-28　优化后的系统阶跃响应曲线

7.3.3 MIMO PI 控制器设计

这是 NCD 模块的第三个示例。运行它，得到闭环系统的方框图，如图 7-29 所示。

Tunable Variables are 2x2 PI Controller gains, Ki and Kp.

1
2
Mux
[T,U]
From Workspace
x' = Ax+Bu
y = Cx+Du
2x2 PI Controller
x' = Ax+Bu
y = Cx+Du
LV100
y
To Workspace
Demux
NCD OutPort 1
NCD_Outport1
NCD OutPort 2
NCD_Outport2
ncd3init

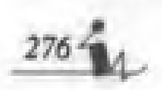
Double click here to initialize plant data and optimization parameters.

Double click on NCD Blockset blocks to display constraint windows.

图 7-29　MIMO PI 控制器方框图

同样的初始化系统，从图 7-29 中可以看出，系统有两个输出，分别连接在一个 NCD 模块上。NCD Outport1 和 NCD Outport2 两个 NCD 模块的优化参数均为 PI 控制器对应的增益矩阵 Kp 和 Ki，如图 7-30 所示。

得到系统的两个输出时域约束图，如图 7-31 和图 7-32 所示。



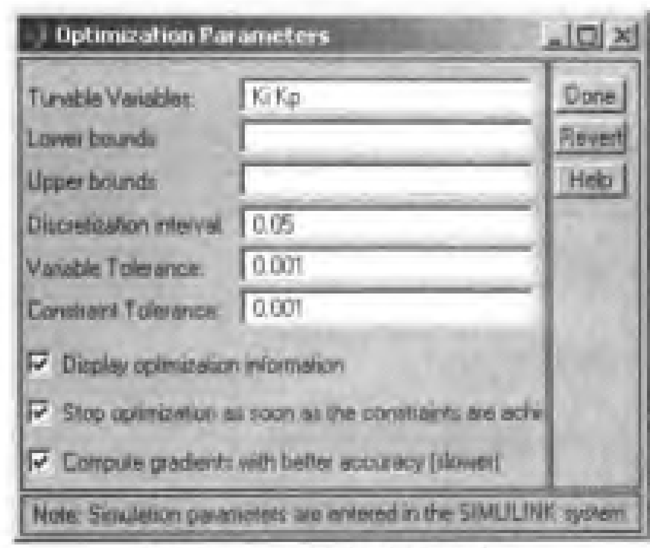


图 7-30 系统优化参数设置窗口

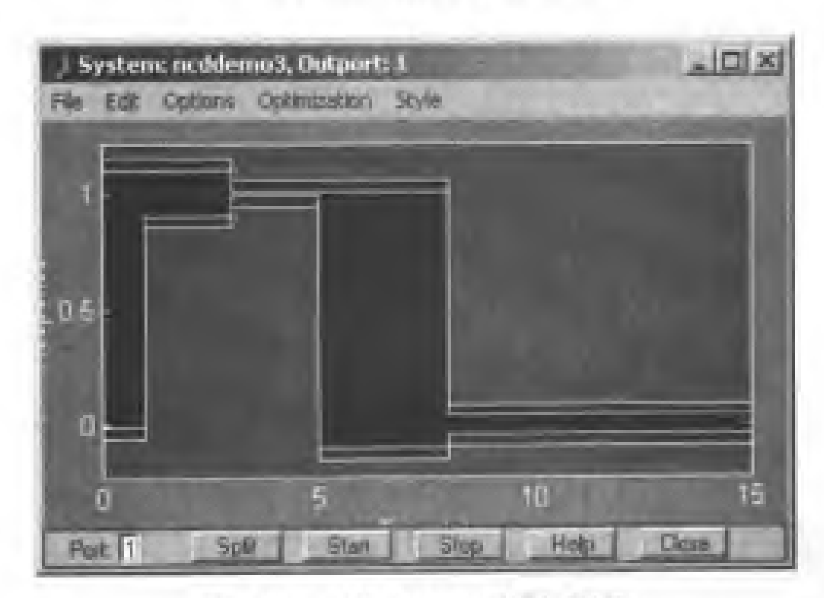


图 7-31 NCD Outport1 的时域约束

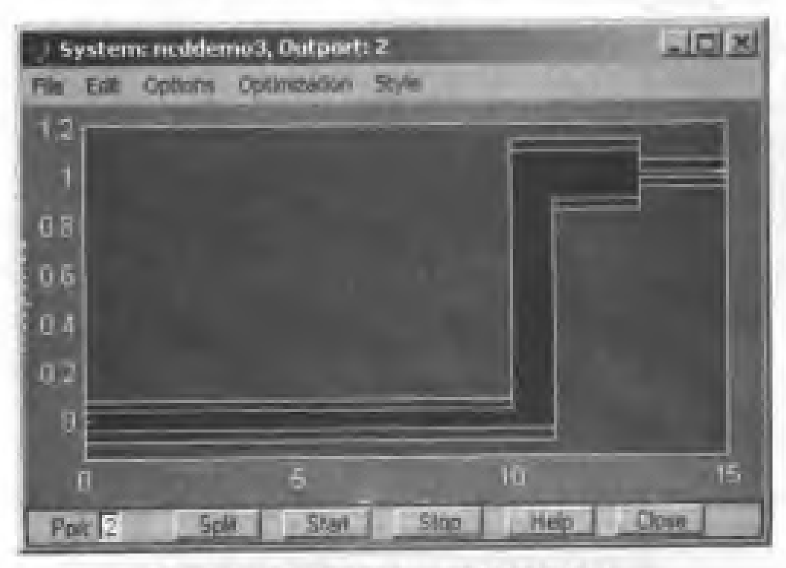


图 7-32 NCD Outport2 的时域约束

设置完不确定变量如图 7-33 所示。

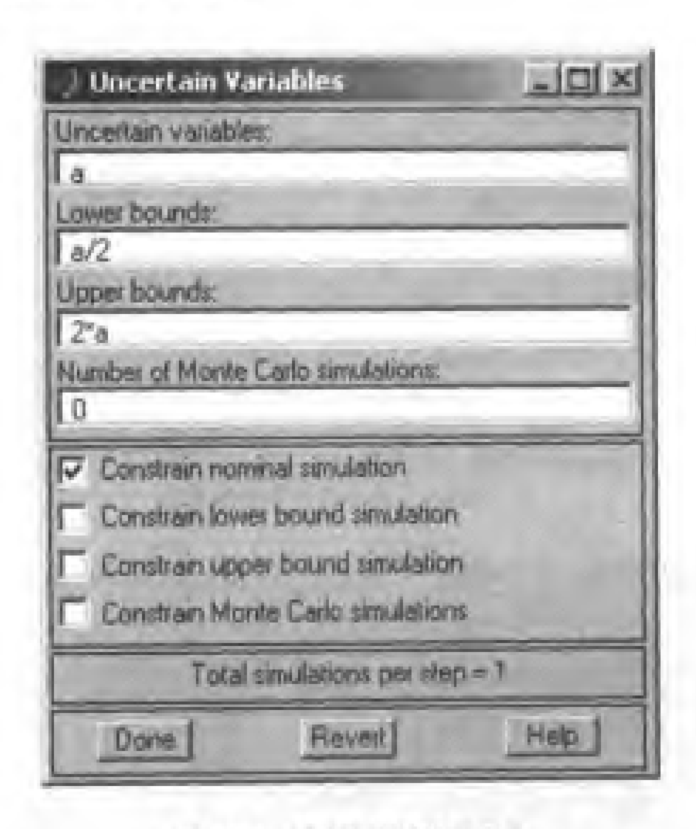


图 7-33 不确定变量设置窗口

对模型进行优化计算，MATLAB 输出结束时系统响应曲线如图 7-34 和图 7-35 所示。

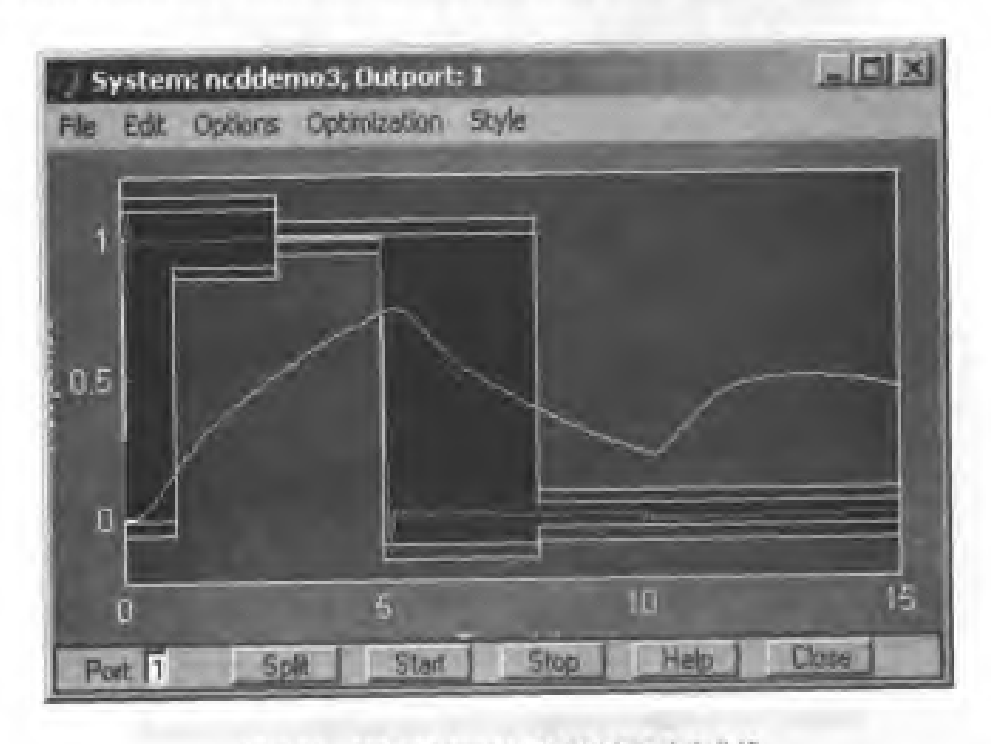


图 7-34 NCD Outport1 时域约束和响应曲线

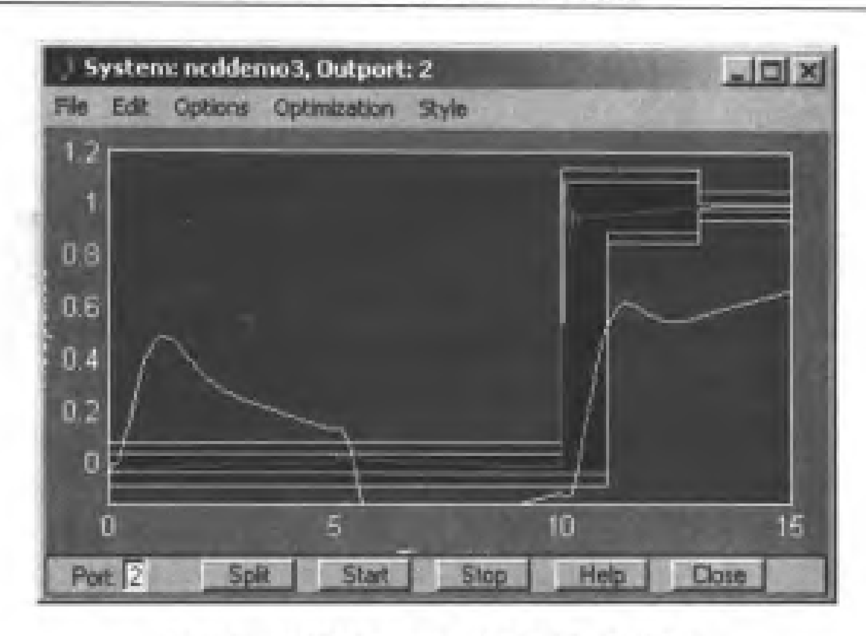


图 7-35　NCD Outport2 时域约束和响应曲线

第8章 控制系统的数学描述

从本章开始，我们正式进入到有关MATLAB在控制系统设计和仿真领域的应用中去。首先来回顾一下控制系统的概念及其发展。本世纪初以来，特别是从第二次世界大战以来，控制科学和控制技术得到了迅速发展。自动控制极大地提高了劳动生产率和产品质量，推动了现代工业的巨大进步。在军事上，控制技术有效地提高了武器的精确度和威力。在航天、制导、核能等方面，控制技术更是不可缺少的。在工业和军事领域中，控制技术的作用是：不需要人的直接参与，而控制某些物理量按照指定的规律变化。

控制理论研究问题是：（1）一个给定的控制系统，它的运动有哪些性质和特征？（2）怎样设计一个控制系统，使它的运动具有给定的性质和特征？前一个问题称为分析，后一个问题称为综合和设计。对于这两个控制理论研究的基本问题，我们将在以后的各章节中详细论述。本章要讨论的问题是，怎样找出适合控制系统的数学描述形式。

众所周知，运动是宇宙存在的普遍形式。我们对控制系统的研究，就离不开对控制系统的运动的研究。所谓运动，并不只是指未知的移动和旋转，而是泛指一切物理量随时间的变化，如温度的升降、电流的强弱、人口的增减等。要研究各种物理量的变化，必须把它们彼此之间相互作用的关系和各自的变化规律用数学形式描述出来。这就是常说的为某一事物建立数学模型。建立描述控制系统运动的数学模型，是控制理论的基础。一般控制系统数学模型的建立都是基于图8-1的思想。

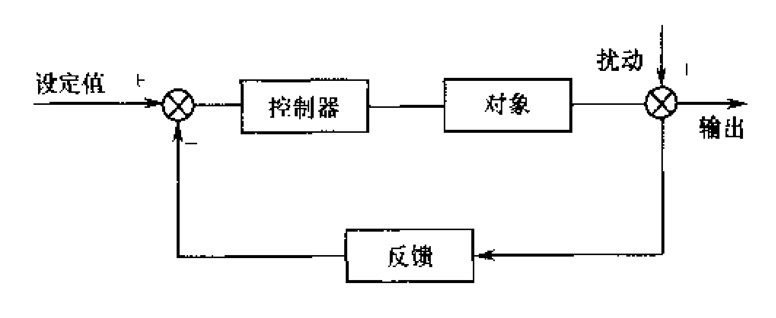


图8-1 控制系统的模型

图8-1是基于偏差控制的反馈控制系统简图。设定值与反馈量的偏差通过控制器作用于被控对象，经过扰动的作用，最终达到控制输出量跟随设定值的目的。对于本图来说，建立数学模型就是要找到一种合适的手段，详细描述被控对象的运动规律，从而为控制器和反馈系数的设计提供可靠的依据。

描述控制系统的数学模型的形式不只一种，它们各有特长和最适用的场合。它们彼此之间也有紧密的联系。大致说来，主要分为两种：一种称为控制系统的时间域描述；另一种称

为控制系统的频率域描述。所谓时间域描述，是指用微分方程和状态方程来描述控制系统，从时间推演的角度刻划系统的运动。而频率域描述，是指用拉普拉斯变换工具来描述控制系统，即通过传递函数从频率变化的角度刻划系统的运动。事实上，这两种数学描述方法的共同基础都是微分方程。状态方程就是一组联立的一阶微分方程，拉普拉斯变换也是在系统微分方程的基础上进行的。因此，本章从系统微分方程的列写和求解入手，详细讨论 MATLAB 在建立系统数学模型方面的应用，并给出大量的实例和一些较为成熟的算法，使读者能够迅速掌握其要领。

8.1　控制系统的运动方程

从工程的角度来说，描写控制系统运动的最基本的数学工具就是微分方程。

要写出描述控制系统运动的微分方程，大致分为两种基本方法。第一种方法是分析系统各部分运动的机理，根据这些机理分别写出描述各部分运动的微分方程，合在一起便成为描述整个系统的方程。第二种方法是人为地在系统上加上某种测试信号，记录系统中各变量的运动，然后选择适当的微分方程，使之能近似地表示这种运动，以此作为系统的方程。有时连测试的信号也不加，就径直记录系统运行时各变量的实际运动，据以建立数学模型。这种方法也称为系统辨识方法，主要用于系统的运动机理复杂而不便分析和不可能分析的情况。

本章的讨论以第一种方法为主，适当兼顾第二种方法，并且主要通过实例来说明。首先我们从线性对象入手，即可以用线性微分方程和线性代数方程描述的对象，并且限定于定常的和集总参数的对象，即其参数不随时间变化，各物理量不随空间位置变化的对象。这种线性、定常、集总参数的对象是工程上最常见的。

8.1.1　微分方程数值解

有关系统微分方程列写的问题，读者可以在任何一本讲述控制理论的教材中找到，其数学基础在工科大学的数学课程中都有所介绍。事实上，除了数学上的熟练程度之外，这还涉及到对所接触的实际问题的理解和系统分析的能力。这并不是本书所能解决的问题，也不是 MATLAB 的所长。因此，除了少数特殊说明的情况之外，本书均假设系统的运动方程或传递函数已经得到。

熟悉信号与系统分析的读者都清楚，得到系统的微分方程之后，通过拉普拉斯变换和反变换，就可以得到线性时不变方程的解析解。对于状态方程，也可以用状态转移矩阵 Φ（t）求解。解析解是精确的，然而通常在计算上存在困难，或根本是不可能的。因此，人们寻找了许多关于微分方程的数值解法，力求在条件允许的范围内逼近方程的解析解。MATLAB 在寻找系统微分方程的数值解方面作的非常出色，MATLAB 提供了三个采用龙格-库塔法（Runge-Kutta）求解微分方程数值解的函数，分别是 ode23（）、ode45（）、ode113（）（4.x 的版本没有 ode113（）函数），对应不同的精度。其中 ode113（）函数精度最高，其基本调用格式为

```
[T,Y] = ODE113('F',TSPAN,Y0,OPTIONS)
```

其中 T 是时间向量，Y 是与 T 相互对应的方程的数值解；'F'是对于系统微分方程的描述，

一般包含在特定的 ODEfile 中；TSPAN 是 1×2 的向量[T0 TFINAL]，表示仿真的开始和结束的时间；Y0 是该微分方程的起始条件；OPTIONS 是控制精度的可选参数，由 odeset（ ）来设置，常用的参数值有'RelTol'，表示 1e-3 精度，或'AbsTol'，表示 1e-6 精度。

单有 ode113（）函数是无法求解微分方程的，必须配合 odefile（ ）函数使用。odefile（ ）函数的功能是描述系统的微分方程和一些相关参数，提供 ode113（ ）函数的接口。其基本的调用格式为：

```
function F = odefile(t,y)
F = < Insert a function of t and/or y here. >;
```

有关系统微分方程的描述，需要说明的是：ode113（ ）函数只接受一阶微分方程的形式。因此，对于高阶微分方程，首先要化为若干个一阶微分方程，然后再使用 odefile（ ）函数。事实上，odefile（ ）函数能够完成的功能很多，例如起始条件的设置、求解该微分方程的雅可比矩阵（Jacobian Matrix）、数值解过零区域的设置等等，都可以提供调用不同格式的 odefile（ ）函数来完成。ode113（ ）函数也有许多衍生的函数。有关这方面的情况就不再一一赘述了，有特殊需要的读者请参阅 MATLAB 帮助。下面以一个简单的例子，使读者熟悉一下 ode113（）函数和 odefile（）函数的基本用法，掌握使用 MATLAB 求解系统微分方程数值解的基本步骤。

某机械运动系统如图 8-2 所示，物体质量 M=1kg，弹簧的虎克系数为常数 K=20N/m，摩擦系数也为常数 B=5N/m/s。t=0 时，施加外力 f（t）=30N，试给出系统的运动方程，何时达到稳态，并画出该机械系统位移、速度随时间变化的坐标曲线以及速度与位移的关系曲线。

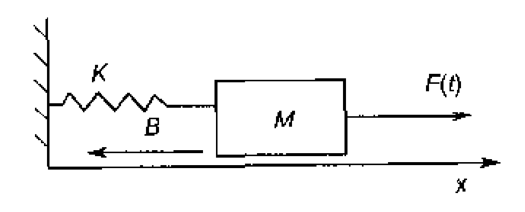


图 8-2　机械系统示意图

结果：根据牛顿运动定理，可以建立系统的运动方程为：

$$M\frac{d^2x}{dt^2}+B\frac{dx}{dt}+Kx=f(t)$$

其中 x 表示物体的位移，其一阶导数和二阶导数分别是物体的速度和加速度。

系统的稳定点：

system values in the stable state:

Time is 2.407091　　Displacement is 1.502046　　Velocity is 0.009417

系统仿真图：

其横坐标为时间，纵坐标为弹簧系统的位移时间响应，如图 8-3 所示。

其横坐标为时间，纵坐标为弹簧系统的速度时间响应，如图 8-4 所示。

其横坐标为弹簧系统的位移，纵坐标为速度，如图 8-5 所示。

求解过程：

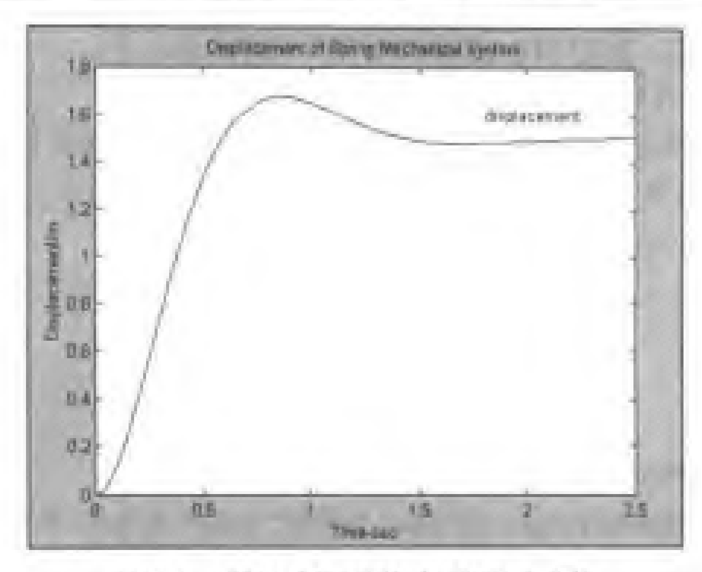


图 8-3　弹簧机械系统位移时间响应曲线

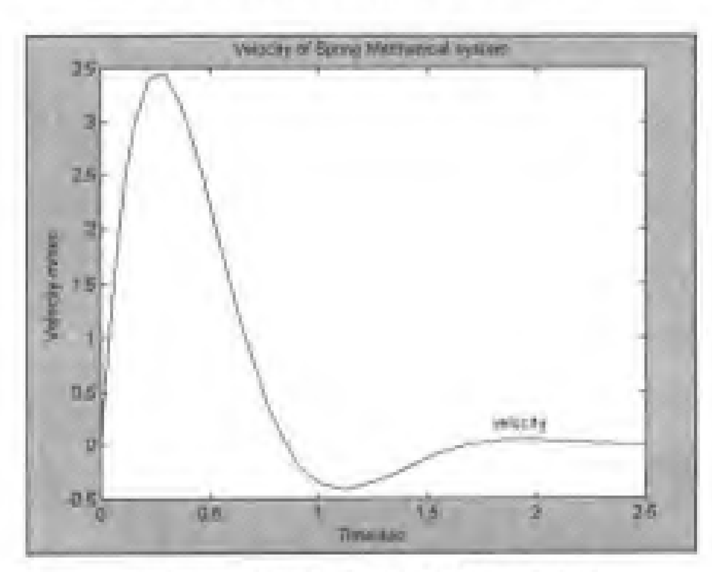


图 8-4　弹簧机械系统速度时间响应曲线

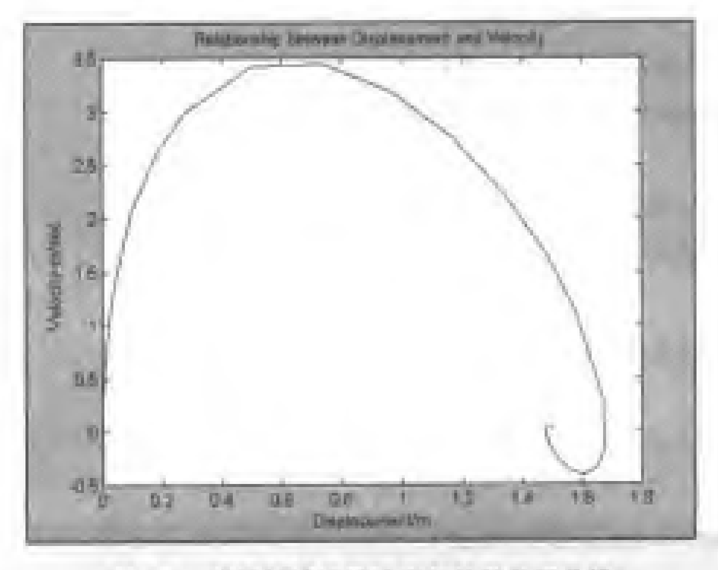


图 8-5　弹簧机械系统速度与位移关系曲线

为得到上图的结果，在 MATLAB 环境下共分三步进行：

1．编写系统微分方程

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 函数“springsys”：

```
function xt = springsys(t,x)
ft=30;
M=1;
B=5;
K=20;
xt=[x(2);1/M*(ft-B*x(2)-K*x(1))];
```

选择 MATLAB Editor/Debugger 的“File”菜单的“Save”选项，保存文件名为“springsys.m”。

2．编写系统的仿真程序

选择“File”菜单新建 M-file，键入以下代码：

```
%关闭所有图形窗口
close all
%数据初始化
ft=30;
M=1;
K=20;
B=5;
t0=0;
tfinal=2.5;
%仿真开始和结束时间
tspan=[t0 tfinal];
%系统初始条件
x0=[0,0];
%设置 ode113（）函数的可选参数
options = odeset('AbsTol',[1e-6;1e-6]);
[t,x]=ode113('springsys',tspan,x0,options);
%系统的加速度
a=1/M*(ft-B*x(:,2)-K*x(:,1));
%寻找系统的平衡点
i=1;
while (abs(a(i))>0.1)|(abs(x(i,2))>0.01)
    i=i+1;
end
%显示寻找结果
disp('system values in the stable state:');
result=sprintf('Time is %f\n',t(i));
```

```
disp(result);
result=sprintf('Displacement is %f\nVelocity is %f',x(i,1),x(i,2));
disp(result);
%绘图
%位移向量
d=x(:,1);
%速度向量
v=x(:,2);
%绘制时间—位移图
plot(t,d);
title('Displacement of Spring Mechanical system');
xlabel('Time-sec');
ylabel('Displacement/m');
text(1.8,1.6,'displacement');
%绘制时间—速度图
plot(t,v);
title('Velocity of Spring Mechanical system');
xlabel('Time-sec');
ylabel('Velocity-m/sec');
text(1.8,0.2,'velocity');
%绘制位移—速度图
plot(d,v);
title('Relationship between Displacement and Velocity');
xlabel('Displacement/m');
ylabel('Velocity-m/sec');
```

选择“File”菜单的“Save”选项，保存文件名为“spring.m”，注意和 springsys.m 保存在同一路径下。

3．运行

选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 spring，在 MATLAB Command Window 下查看运行的结果。

分析：如前所述，ode113（）函数只能求解形如 $y'=F(y,t)$ 格式的一阶微分方程的数值解。因此，需要将本题的二阶微分方程化为两个一阶微分方程的格式，然后再求解。根据系统的运动方程：

$$M\frac{\mathrm{d}^2x}{\mathrm{d}t^2}+B\frac{\mathrm{d}x}{\mathrm{d}t}+Kx=f(t)$$

令
$$x_1=x,x_2=\frac{\mathrm{d}x}{\mathrm{d}t}$$

有
$$\frac{\mathrm{d}x_1}{\mathrm{d}t}=x_2$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t}=\frac{1}{M}\left(f\left(t\right)-Bx_2-Kx_1\right)$$

这样，就可以应用 ode113 函数对这两个一阶微分方程进行仿真了。

关于 odefile（）函数的使用，需要说明的是它与其他函数不同，odefile（）其实是一种编写M函数的格式，不过这种M函数只能配合ode函数使用。读者仔细阅读上文的springsys.m文件的代码，就可以理解 odefile（）函数的使用方法。

通过图 8-5 的系统速度与位移的关系曲线可以看出，系统有一个平衡点，并且是经过小幅振荡后逐渐趋于平衡点的。从其他仿真结果来看，系统最终的平衡点在距原点 1.5m 处左右，此时速度和加速度均接近于零。当然，严格说来速度和加速度都不可能达到零，但在工程上进入平衡区域上下限的 5%以内就可以认为达到稳态了。因为系统是从零状态出发的，并且有过零的区域，所以判断是否达到稳态时使用了加速度和速度的双重判据，只有在速度和加速度均接近于零时才判定达到稳态。从控制理论的角度来看，如果把稳态的位移看作希望加以控制的输出量，那么物体、虎克系数和摩擦系数就组成被控对象的固有特性，外力 $f(t)$ 就是控制手段。通过施加不同的外力就可以期望得到不同的稳态位移。当然，这只是开环控制，抗干扰的能力很差，例如虎克系数或摩擦系数的微小变化就会造成平衡点的较大偏移。如果希望得到较好的系统性能和抗干扰能力，可以把当前位移与期望位移的偏差以电信号或别的形式反馈给施加外力的机构，从而形成本章开始提出的按偏差控制的反馈控制系统模型。

小结：这是我们第一次接触一个完整的控制系统的例子，因此讲述的比较详细。包括 M 函数及文件的编写和调试、运行、存盘等。当然，这个例子还是比较简单的，程序也不是很长，结构和解题思路也都比较清晰，读者比较容易接受。除了 ode113（）函数之外，ode23（）、ode45（）、ode15s（）、ode23s（）、ode23t（）、ode23tb（）等函数也可以用来进行系统微分方程的仿真，并且能够满足不同的需求。程序中使用了一些以前没有涉及的 MATLAB 函数，例如“sprintf”等等，一方面因为不是解题的重点，另一方面这些函数的含义也比较容易理解，所以其具体使用方法就不在这里赘述了，读者可以自己查阅相关的 MATLAB 帮助。以后的例子也是采取这种方式，当然，如果是比较关键或难于理解的函数，还是会适当解释的。

8.1.2 非线性系统描述

严格说来，所有的控制系统都是非线性的。包括上一节分析的弹簧机械系统，当位移和速度变化时虎克系数和摩擦系数也会产生变化，而且这种变化是位移和速度的函数，这就在系统的微分方程中引入了非线性的因素。但是我们为什么要用线性的模型来分析它呢？这是因为在一定的范围内，这些物理量的非线性变化非常小，按照线性的模型进行控制是精度所允许的，与实际的系统吻合的非常好。并且，系统的线性模型是我们进行理论研究的最重要的手段。这就像物理学的质点和信号处理中白噪声的概念一样，虽然在现实世界中是不存在的，但它们是整个理论研究的基石。

还有一些系统，非线性的因素占据主导地位。对于这样的系统，首先要给出其非线性的模型，然后再进行仿真或进行线性化处理，否则是无法揭示系统的本质的。因此，研究非线性系统的运动是非常必要并且十分重要的。描述非线性系统比较传统的方法有描述函数法、相平面法和波波夫法（Popov）等，这些方法的特点都是以线性模型来近似非线性模型或是直观的描述，作为理论分析是很有用处的。但由于数字式计算机的兴起，除了定性的分析之外，

这些方法在工程实际中已经很少使用了。事实上，我们可以对非线性模型直接仿真。MATLAB 提供的 ode 系列函数同样可以进行非线性系统仿真，其调用方式和格式与线性系统完全一样，请看下例：

如图 8-6 所示的单摆，绳长 $L=1\text{m}$，物体质量 $M=1\text{kg}$，运动阻尼系数 $B=0.1\text{kg/m/s}$。试给出系统的运动方程，并画出系统位移、角速度随时间变化以及角速度与位移的关系图。

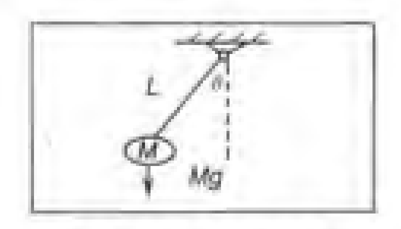


图 8-6　单摆系统示意图

结果：系统的运动方程为

$$ML\ddot{\theta} + BL\dot{\theta} + Mg\sin\theta = 0$$

系统的仿真结果如图 8-7 所示。

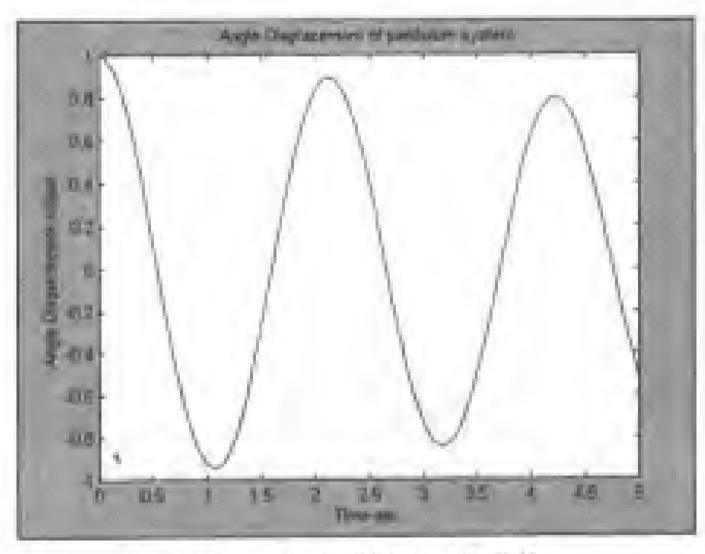


图 8-7　单摆系统位移响应曲线

其横坐标为时间，纵坐标为单摆系统的角位移时间响应，如图 8-7 所示。从图 8-7 中的响应曲线的类型来看，是代阻尼的三角函数型，围绕原点左右振荡，并随时间的递增振荡幅度逐渐减小（这一点不太明显，不过仔细观察图 8-7 可以发现响应曲线第一个峰值约为 0.9，第二个峰值则约为 0.85，逐渐衰减）。

其横坐标为时间，纵坐标为单摆系统的角速度时间响应，如图 8-8 所示。同样，响应曲线也是代阻尼的三角函数型，这是因为速度是位移的微分。

其横坐标为单摆系统的位移，纵坐标为角速度，如图 8-9 所示。

为得到上图的结果，在 MATLAB 环境下共分三步进行：

1. 编写系统微分方程

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 函数 “pendulum”：

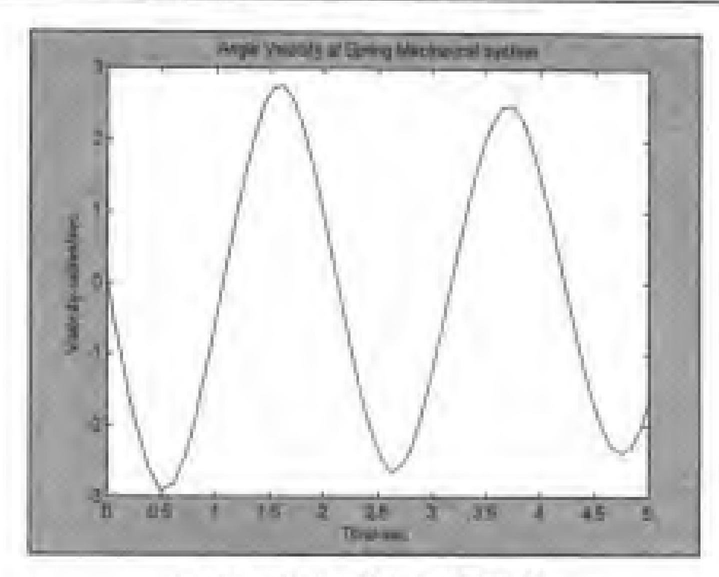


图 8-8 单摆系统速度响应曲线

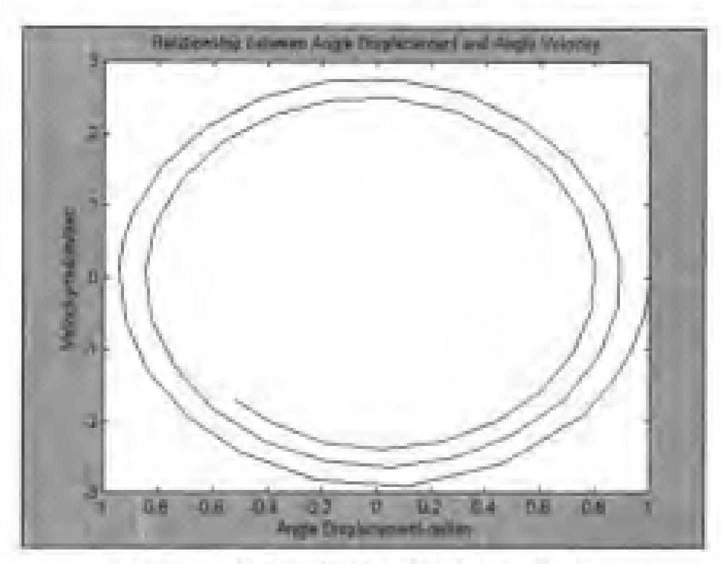


图 8-9 单摆系统位移与速度关系曲线

```
function xt = pendulum(t,x)
g=9.81;
M=1;
B=0.1;
L=1;
xt=[x(2);-B/M*x(2)-g/L*sin(x(1))];
```

选择 MATLAB Editor/Debugger 的"File"菜单的"Save"选项，保存文件名为"pendulum.m"。

2. 编写系统的仿真程序

选择 "File" 菜单新建 M-file，键入以下代码：

```
%关闭所有图形窗口
close all
%数据初始化
tfinal=5;
```

```
%仿真开始和结束时间
tspan=[t0 tfinal];
%系统初始条件
x0=[1,0];
%设置 ode113（）函数的可选参数
options = odeset('AbsTol',[1e-6;1e-6]);
[t,x]=ode113('pendulum',tspan,x0,options);
%绘图
%位移向量
theta=x(:,1);
%速度向量
av=x(:,2);
%绘制时间—位移图
plot(t,theta);
title('Angle Displacement of pendulum system');
xlabel('Time-sec');
ylabel('Angle Displacement-radian');
%绘制时间—速度图
plot(t,av);
title('Angle Velocity of Spring Mechanical system');
xlabel('Time-sec');
ylabel('Velocity-radian/sec');
%绘制位移—速度图
plot(theta,av);
title('Relationship between Angle Displacement and Angle Velocity');
xlabel('Angle Displacement-radian');
ylabel('Velocity-radian/sec');
```

选择“File”菜单的“Save”选项，保存文件名为“simupendu.m”，同样要注意和 pendulum.m 保存在同一路径下。

3. 运行

选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 simupendu，在 MATLAB Command Window 下查看运行的结果。

分析：与前述的弹簧机械系统相同，本例的系统运动方程也是二阶的微分方程。因此，首先要化为两个一阶的微分方程，然后再进行仿真。根据系统的运动方程：

$$ML\ddot{\theta} + BL\dot{\theta} + Mg\sin\theta = 0$$

令
$$x_1 = \theta, x_2 = \dot{\theta}$$

有
$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = x_2$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t}=-\frac{B}{M}x_2-\frac{g}{L}\sin x_1$$

这两个一阶微分方程就是系统的状态方程，这里选择的两个状态分别是单摆的角位移和角速度。

从仿真结果来分析，单摆的角位移和角速度都是过零的正弦曲线，符合事先对其受力的分析。从图 8-9 可以看出，系统的角位移与角速度的关系曲线是一支内旋的螺旋曲线，逐渐向原点方向旋转。这是由于阻尼系数 B 的存在，使得单摆的动能逐渐减小，最终动能为零，单摆回到原点。如果阻尼系数为零，我们将看到一个中心在原点的椭圆（或圆）。

事实上，类似图 8-9 这种系统的位移与速度曲线就是其相平面图，从图上可以判断系统的运动趋势、稳定性、有无平衡点等等一系列性质。当然，这种方法不如直接仿真来得精确和直观。

对于非线性系统，还可以按照 Taylor 展开的方法直接将其线性化，根据精度来控制展开的阶数。对于本例，设 $\theta=\theta_0+\Delta\theta$，则系统的运动方程如下：

$$ML\left(\ddot{\theta}_0+\Delta\ddot{\theta}\right)+BL\left(\dot{\theta}_0+\Delta\dot{\theta}\right)+Mg\sin\left(\theta_0+\Delta\theta\right)=0$$

当 $\Delta\theta$ 趋近于零，又因为单摆的振荡幅度较小，有：

$$\sin\theta_0\approx\Delta\theta \qquad \sin\Delta\theta\approx 0,\cos\Delta\theta\approx 1$$

因此，根据三角公式将系统的运动方程展开，有：

$$ML\Delta\ddot{\theta}+BL\Delta\dot{\theta}+Mg\Delta\theta=0$$

感兴趣的读者可以对这个线性微分方程进行仿真。通过仿真结果就会发现，在满足上文假设的条件时，其结果与根据非线性方程仿真的结果基本一致。当然，如果单摆的振荡幅度很大，上述的线性方程就不再成立，只能用非线性方程进行仿真了。

小结：本例通过对非线性的单摆系统进行仿真，简要介绍了对非线性系统进行数学描述的方法。事实上，读者可能已经发现了，线性系统仿真和非线性系统仿真在使用 ode113( ) 函数的格式和用法上并无任何不同，区别仅仅在有关系统运动方程函数的编写上。因此，在使用 MATLAB 进行控制系统设计和仿真时，在大多数情况下不必刻意区分线性还是非线性系统，实际操作起来都是一样的。

需要注意的是，一般说来，非线性系统对于仿真的精度是比较敏感的，尤其是在系统稳定性和抗干扰能力的问题上，有可能在不同的仿真精度下得到不同的结果。因此，对于实际接触非线性系统，首先要进行理论分析，选择合适的模型和精度，然后再用 MATLAB 进行仿真。

有关系统的微分方程描述就介绍到这里，它是控制系统数学描述的基础，在以后的论述中还要经常提及。

8.2 控制系统的传递函数描述

上一节中我们讨论的是在时间域内描述动态系统的方法，即直接以微分方程为工具进行研究的方法。从这一节开始，我们要叙述另一种数学描述方法。这种方法不是直接去求解和

讨论微分方程本身，而使用拉普拉斯变换建立一种数学模型，称为传递函数；用传递函数来研究对象的运动。在这种方法中，自变量不是时间，而是拉普拉斯变换中的复数变量 s，称为复频率。所以这种建筑在拉普拉斯变换和传递函数基础上的描述方法又称为频率域方法。简单地说，时间域描述方法以时间 t 为自变量，频率域描述方法以复频率 s 为自变量。

线性时不变系统（Linear Time Invariant System，简称为 LTI 系统）的传递函数的定义为零初值条件下输出量的拉普拉斯变换与输入量的拉普拉斯变换像函数之比。尽管传递函数只能用于线性系统，但它比微分方程提供更为直观的信息。令传递函数的分母多项式为零，便得到系统的特征方程。特征方程的根是系统的极点，分子多项式的根是系统的零点。那么传递函数便可由常数项与系统的零、极点决定。利用传递函数，我们可以方便地研究系统参数的改变对系统响应的影响。通过拉普拉斯反变换可以得到系统的时域响应，这通常需要用到函数的部分分式展开。

通常有两种方法可以得到系统的传递函数。一种是根据定义，首先列写系统的微分方程，然后对该方程两边同时进行拉普拉斯变换，通过输出量拉普拉斯变换与输入量拉普拉斯变换像函数之比得到系统的传递函数。如果系统是线性的，那么传递函数就不因输入量函数或输出量函数的改变而改变。另一种是实验的方法，首先大致估计一下系统的阶次，然后加典型的测试信号，根据系统的响应曲线来求得相关的参数，最终获得系统的传递函数模型。这种方法一般适用于系统的机理比较难于分析，并且阶次不高的情况。当然，用这种方法所求得模型的误差也是比较大的。限于篇幅，本书所讨论的主要是第一种方法。在以下的叙述中，如果不作特殊说明，均假设系统的微分方程已知。

MATLAB 提供了许多功能强大的内部函数可以构建和处理系统的传递函数。包括多项式求根、求解传递函数的零极点和增益、传递函数部分分式展开等等。能够非常方便快捷地建立起系统的传递函数模型。

8.2.1 传递函数的零点和极点

设某控制系统只有一个输入量 u（t），只有一个输出量 y（t）。并设这个系统可以用线性微分方程：

$$a_n\frac{\mathrm{d}^n y}{\mathrm{d}t^n}+a_{n-1}\frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}+\cdots+a_1\frac{\mathrm{d}y}{\mathrm{d}t}+a_0 y=$$

$$b_m\frac{\mathrm{d}^m u}{\mathrm{d}t^m}+b_{m-1}\frac{\mathrm{d}^{m-1} u}{\mathrm{d}t^{m-1}}+\cdots+b_1\frac{\mathrm{d}u}{\mathrm{d}t}+b_0 u$$

描述，其中 $n\geqslant1$，$m\geqslant1$，多项式首项不为零。假设 u（t）和 y（t）及其各阶导数在零负时刻的初值均为零。对上式两边同时取拉普拉斯变换，有

$$a_n s^n\overline{y}(s)+a_{n-1}s^{n-1}\overline{y}(s)+\cdots+a_1 s\overline{y}(s)+a_0\overline{y}(s)=$$

$$b_m s^m\overline{u}(s)+b_{m-1}s^{m-1}\overline{u}(s)+\cdots+b_1 s\overline{u}(s)+b_0\overline{u}(s)$$

令：

$$G(s)=\frac{\overline{y}(s)}{\overline{u}(s)}$$

有：

$$G(s)=\frac{b_m s^m+b_{m-1}s^{m-1}+\cdots+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+\cdots+a_1 s+a_0}$$

此 $G(s)$ 就称为系统的传递函数。这也就是上文提到的第一种由微分方程得到系统传递函数的方法。

从纯数学的角度来看，传递函数就是两个有理多项式之比。传递函数的分母多项式称为系统的特征多项式，此多项式的根称为系统的极点。传递函数的分子多项式的根称为系统的零点。对于 LTI 系统来说，系统的所有信息都包含在其传递函数的极点和零点之中。研究传递函数，首先就要研究其极点和零点。

对于上文提到的只有一个输入量和一个输出量的系统，称为 SISO 系统（Single-Input Single-Output）。在 MATLAB 环境下可以方便地用两个行向量表示 SISO 系统的传递函数，分别代表其分子和分母的系数：

$$\mathrm{num}=\left[b_m,b_{m-1},\cdots,b_1,b_0\right] \qquad \mathrm{den}=\left[a_m,a_{m-1},\cdots,a_1,a_0\right]$$

两个行向量的元素分别是原 LTI 系统传递函数分子和分母多项式系数的降幂排列。这两个行向量可以取不同的名字。但在 MATLAB 环境下对控制系统进行仿真时，习惯用 num 和 den 来命名，这样可读性较强。

有了描述系统传递函数的行向量之后，就可以求取其零点和极点了。MATLAB 提供了一条 tf2zp（）函数，可以用来根据系统的传递函数求取其零点、极点和增益。其基本调用格式如下：

```
[Z,p,k] = TF2ZP(NUM,den)
```

其中 Z 和 p 是行向量（对于一阶系统来说是标量），包含了系统的零点和极点。k 是标量，表示系统的增益。也就是说，tf2zp（）函数将原系统的传递函数化为如下的形式：

$$G(s)=k\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}$$

其中 $$Z=\left(z_1,z_2,\cdots,z_m\right),\quad p=\left(p_1,p_2,\cdots,p_n\right)$$

根据系统的传递函数，令 $s=j\omega$，还可以求取系统的频率响应，从而对系统的性能作出评价。MATLAB 提供了一条函数 freqs（），可以根据系统的传递函数求取其频率响应数据。其基本调用格式为：

```
H = FREQS(B,A,W)
```

其中 B 和 A 是系统传递函数的分子和分母多项式，W 是频率采样点；H 是行向量，其元素是对应采样点 W 的系统复频率响应。下面通过一个简单的例子，简要介绍一下这些函数的基本用法。

考虑前面例子中的弹簧机械系统，其系统运动方程如下：

$$M\frac{\mathrm{d}^2x}{\mathrm{d}t^2}+B\frac{\mathrm{d}x}{\mathrm{d}t}+Kx=f(t)$$

其中 $M=1\mathrm{kg}$，$K=20\mathrm{N/m}$，$B=5\mathrm{N/m/s}$。仅在 $t=0$ 时，施加外力 $f(t)=1\mathrm{N}$。试给出系统的传递函数，求解其零点、极点和增益，并绘制其复平面直角坐标的 Nyquist 图和系统的频率响应曲线。

结果：系统的传递函数为： $$G(s)=\frac{1}{s^2+5s+20}$$

系统的零点、极点和增益分别为：

system zero-points are　　z =Empty matrix: 0-by-1

system polar-points are

p = -2.5000 + 3.7081i　-2.5000 - 3.7081i

system gain is　　　　k = 1

系统仿真结果如图 8-10 所示。

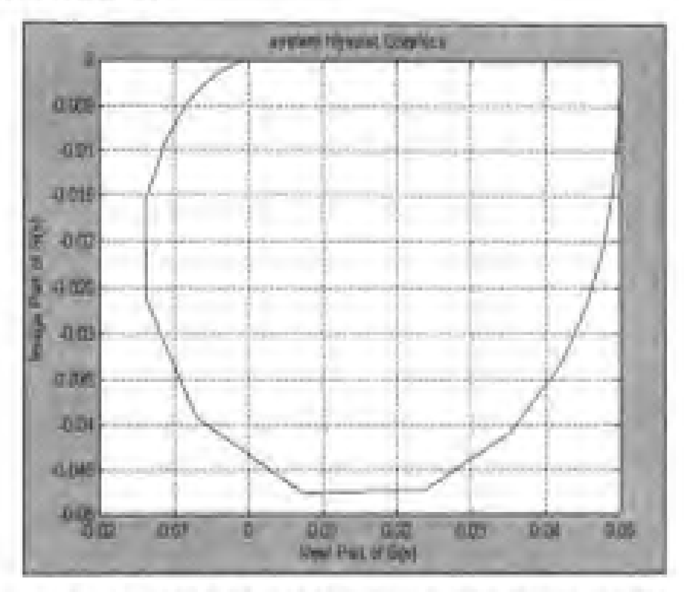


图 8-10　弹簧机械系统复平面直角坐标 Nyquist 曲线

分析：本例的主要目的是分析系统传递函数本身的性质，因此取输入函数 $f(t)=\delta(t)$。当然，系统的频率特性曲线的形状和稳定性与输入量的选取无关。从 tf2zp（）函数输出的结果来看，系统没有零点，这与原传递函数分子多项式为常数项是相吻合的。系统有两个极点 -2.5+3.7801i 和-2.5-3.7801i，实部小于零，极点均在复平面的左半平面，说明系统是稳定的。从系统的复平面直角坐标 Nyquist 图来看，系统的 Nyquist 曲线并不包围（-1，0）点。右半平面零点数为零，根据 Nyquist 稳定判据，系统是稳定的。这同前边对系统极点的分析得出的结果是一致的。

其横坐标为系统频率响应的实部，纵坐标为系统频率响应的虚部。如图 8-11 所示。

对照前例中系统的时间响应曲线，系统最终将趋于一个平衡点，因此系统是稳定的。系统的 Nyquist 曲线是半个心脏线的形状，这是因为对频率 ω 采样的过程中只选取了（0，+∞）的点。如果包含（-∞，0）的频率点，那么系统的 Nyquist 曲线应该是一个完整的心脏线的形状。不过对于所有的 LTI 系统来说，其 Nyquist 曲线均关于实轴对称，仅选取（0，+∞）的频率点就足以看出曲线的形状和变化趋势了。

在绘制系统的频率响应曲线时使用了 x 轴的半对数坐标系，横轴是对数频率，纵轴分别是传递函数频率响应的模和相角。从系统的幅频响应曲线来看，大约在 ω=4 左右的地方有一个谐振峰。谐振峰越大，系统的响应曲线振荡越剧烈。也就是说，系统对该频率的输入信号响应最为敏感。本系统的谐振峰比较小，说明系统的运动比较平缓。从前例系统的时间响应曲线来看，只经过一次振荡就趋于稳态值，验证了上面的说法。从系统的相频响应曲线来看，相角趋于-180°（-π）时传递函数的模已趋于零，与 1 相差很远。因此，系统有较大的稳定裕量，存在充分的空间可以增加控制手段，使其输出量跟随期望的设定值。

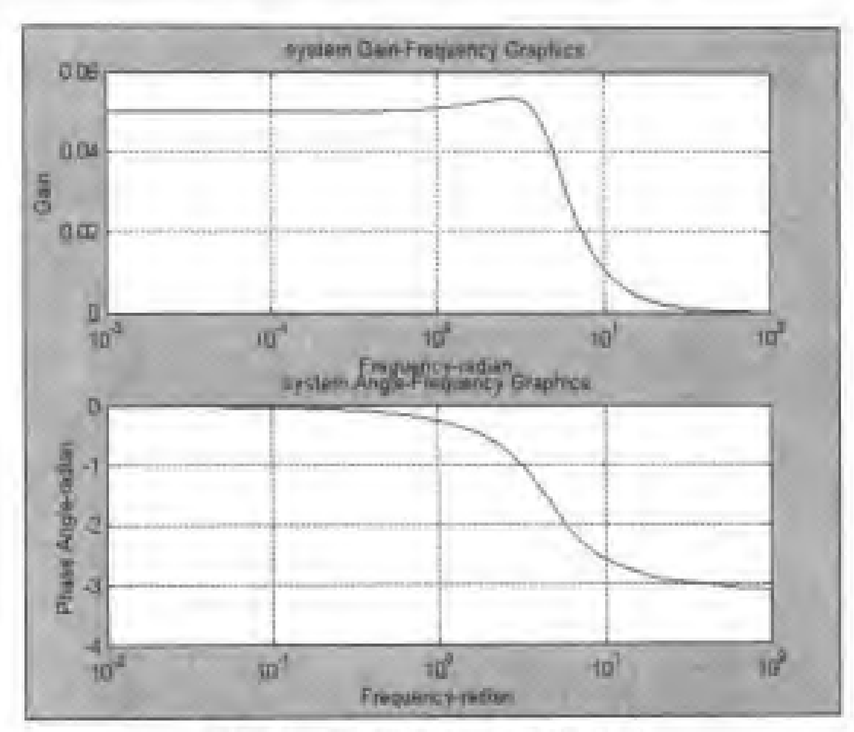


图 6-11 弹簧机械系统频率响应曲线

求解过程：

本例的解题步骤分为三部分。

1．求取系统的传递函数

对系统的运动方程两边取拉普拉斯变换，得：

$$Ms^2\overline{x}(s)+Bs\overline{x}(s)+K\overline{x}(s)=\overline{f}(s)$$

$$G(s)=\frac{1}{Ms^2+Bs+K}=\frac{1}{s^2+5s+20}$$

2．求解系统的零点、极点和增益并绘制系统频率响应曲线

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "springfreq.m"：

```
%关闭所有图形窗口
close all;
%数据初始化
%传递函数分子多项式
num=1;
%传递函数分母多项式
den=[1 5 20];
%求系统零点、极点和增益
w=logspace(-2,2);
[z,p,k]=tf2zp(num,den);
%显示结果
disp('system zero-point is');
```

```
%系统零点
z
disp('system polar-point is');
%系统极点
p
disp('system gain is');
%系统增益
k
%求系统的频率响应
H=freqs(num,den,w);
x=real(H);
y=imag(H);
%图形绘制
%复平面直角坐标 Nyquist 图
plot(x,y);
title('system Nyquist Graphics');
xlabel('Real Part of G(s)');
ylabel('Image Part of G(s)');
grid;
subplot(211),
%传递函数增益
q=abs(H);
semilogx(w,q);
title('system Gain-Frequency Graphics');
xlabel('Frequency-radian');
ylabel('Gain');
grid;
%传递函数相角
alpha=angle(H);
subplot(212),
semilogx(w,alpha);
title('system Angle-Frequency Graphics');
xlabel('Frequency-radian');
ylabel('Phase Angle-radian');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“springfreq.m”。

3．运行

选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 springfreq，在 MATLAB Command Window 下查看运行的结果。

小结：本例从频率域的角度重新分析了前例介绍的弹簧机械系统，所得到的结果与原先的结论是完全一致的。

应该说，时间域分析方法和频率域分析方法各有所长，各自有其适用的领域。时间域分析方法的特点是非常直观，所有想要的信息一目了然，分析结果与直接应用紧密结合，并且适用于各种线性和非线性系统。

频率域分析虽然是间接的方法，但对于线性系统分析来说，所有信息都包含在其传递函数之中了。并且，人们总结了大量关于频率域分析的规律和定理，直接应用这些规律和定理比时间域分析方法方便快捷，能收到事半功倍的效果。例如判断系统的稳定性时，只需求解系统极点的实部即可，不必绘制其时间响应曲线。

对于控制领域的专业人员来说，虽然数字式计算机包括类似 MATLAB 等仿真软件的出现，使得时间域分析同样简便易行，但同时熟练掌握这两种分析方法还是非常必要的。它们互为表里，相互补充。

在上文的程序中使用了一个以前没有接触过的函数 logspace（），其调用格式为 LOGSPACE(d1, d2, N)，功能是产生 N 个对数等间距的采样点，范围是 10 的 d1 次幂到 10 的 d2 次幂之间。如果不给出参数 N，则产生 50 个对数等间距的采样点。细心的读者可以发现，因为对数函数的限制，这里的采样点均大于零。事实上，这也是为什么上文系统的复平面直角坐标 Nyquist 图只绘制了实轴以下部分的原因。

对于离散控制系统来说，其运动方程是以差分方程的形式描述的。相应地对其差分方程两边同时进行 Z 变换，就可以得到该离散控制系统的脉冲传递函数。脉冲传递函数在 MATLAB 中的实现方法和处理手段与 s 域的传递函数基本一致，也是用两个行向量来代表脉冲传递函数的分子和分母多项式。只是在分析过程中要注意到函数的自变量由 s 域转化为 Z 域，s 域的左半平面转化为 Z 域的单位圆内部。

传递函数还可以描述多输入多输出的系统（Multi-Input Multi-Output，简称 MIMO 系统），这就引出了传递函数矩阵的概念。传递函数矩阵的描述方法很多，英国学派的多变量频域设计方法就是基于系统的传递函数矩阵描述的。限于篇幅，对于脉冲传递函数和传递函数矩阵这里就不再介绍了，以后接触到相关的概念时会作一些初步的解释。感兴趣的读者请参阅相关的资料。

8.2.2 传递函数的部分分式展开

根据传递函数系统的时间域响应时，通常要用到有理分式的部分分式展开。所谓部分分式展开，就是将高阶的有理分式化为若干个一阶有理分式之和的形式。如果传递函数 G（s）不包含多重极点，那么，将 $G(s)$ 用部分分式展开后即可得到：

$$G(s)=k+\sum_{i=1}^{n}\frac{r_i}{s-p_i}$$

k 是常数项，对于真分式来说 $k=0$。r 是各分式的系数，p 是系统的极点。

对上式求拉普拉斯逆变换，可得系统的冲激响应为：

$$y(t)=k\delta(t)+\sum_{i=1}^{n}r_i\,\mathrm{e}^{p_i t}$$

如果要求解系统在输入函数 $u(s)$ 下的时间响应，只需对 $G(s)\cdot u(s)$ 进行部分分式展开并求拉普拉斯逆变换即可。

MATLAB 提供了一条函数 residue（）可以求解有理分式的部分分式展开，其基本调用格式为：

[R,P,K] = RESIDUE(B,A)

其中 B 和 A 分别表示降幂排列的该有理分式的分子和分母多项式系数；R 是求得的部分分式展开的各分式系数，P 是系统极点，K 是常数项。有了这些结果，就可以根据部分分式展开求解系统的时间响应了。

根据系统传递函数，试求取系统的部分分式展开，并绘制系统的冲激响应和阶跃响应。

系统传递函数为：
$$G(s)=\frac{1}{s^2+5s+20}$$

结果：传递函数的部分分式展开系数为：

```
system Fraction coefficients are
r =       0 - 0.1348i
          0 + 0.1348i
system polar-points are
p =      -2.5000 + 3.7081i
         -2.5000 - 3.7081i
system Direct Term is
         k =  [ ]
```

表达式为：

$$G(s)=\frac{-0.1348i}{s+2.5-3.7801i}+\frac{0.1348i}{s+2.5+3.7801i}$$

系统的冲激响应曲线如图 8-12 所示。

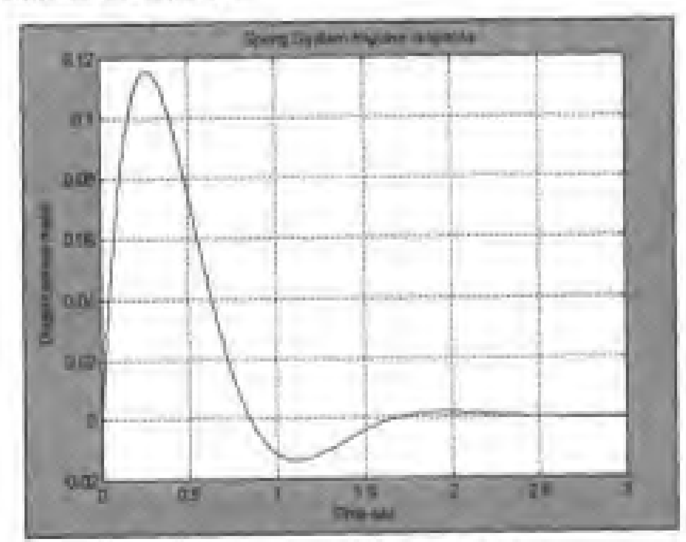


图 8-12 弹簧系统的冲激响应曲线

其横坐标为时间，纵坐标为响应值。

系统的阶跃响应曲线如图 8-13。

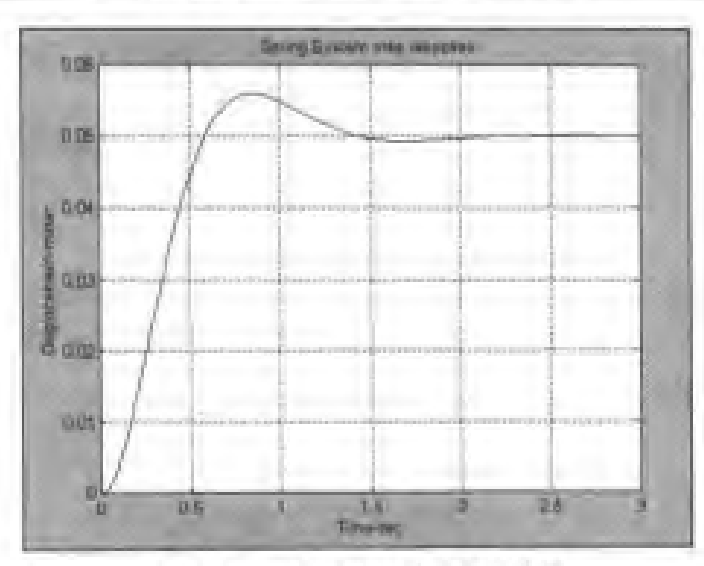


图 8-13 弹簧系统的阶跃响应曲线

分析：residue（）函数的返回值都是以向量的形式存在的。因此，“k=[ ]”表示 k 的值为零。从系统传递函数部分分式展开的结果来看，其极点均为复数。对于二阶系统来说，这表示系统的时间响应是振荡型的。系统的冲激响应就是输入函数为 $\delta(t)$ 的系统时间响应。因为 $\delta(t)$ 的拉普拉斯变换是 1，所以事实上就是系统传递函数的时间响应。

系统的时间域响应曲线是图 8-12，对应的频率域曲线是前例的图 8-11。从图 8-12 可以看出，系统的冲激响应在 T=0.25s 左右达到峰值，对应于图 8-11 的 ω= 4 左右的谐振峰。系统的阶跃响应是输入函数为 1$u(t)$ 时系统的时间响应，其响应曲线是图 8-13。可以看出，图 8-13 与图 8-3 弹簧系统的位移时间响应曲线的形状完全一致，仅仅是纵坐标的刻度不同。事实上，图 8-3 是系统的输入函数为 30 $u(t)$ 时的响应曲线，与图 8-13 在纵坐标的刻度上相差 30 倍。细心的读者还会发现，图 8-12 与图 8-4 的曲线形状也完全一致，仅仅是纵坐标相差 30 倍。但这并不表明图 8-12 也是速度的时间响应曲线。这是因为图 8-4 是速度的时间响应曲线，速度是位移的导数，将该曲线积分就可以得到位移的时间响应曲线。而 $\delta(t)$ 正式 $u(t)$ 的导数，对于 LTI 系统来说，导数和积分都是线性运算，是可以传递的。因此，将系统的冲激响应曲线积分就可以得到系统的阶跃响应曲线。所以，图 8-12 仍是位移的时间响应曲线。

求解过程：

本例的解题步骤分为两部分：

1．求取系统传递函数的部分分式展开和冲激响应

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件“springimp.m”：

```
%关闭所有图形窗口
close all;
%求取传递函数部分分式展开
num=1;
den=[1 5 20];
[r p k]=residue(num,den);
disp('system Fraction coefficients are');
```

```
%各分式系数
r
disp('system polar-points are');
%系统极点
p
disp('system Direct Term is');
%常数项
k
%求解系统的时间响应
t=0:0.01:3;
q=size(t);
m=length(k);
n=length(r);
s=zeros(q);
%判断是否有常数项
if m~=0
    s(1)=k;
end
for i=1:n
   s=s+r(i)*exp(p(i)*t);
end
%绘图
plot(t,s);
title('Spring System Impuls response');
xlabel('Time-sec');
ylabel('Displacement-meter');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“springimp.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 springimp，在 MATLAB Command Window 下查看运行的结果。

2、求取系统的阶跃响应曲线

在 MATLAB Editor/Debugger 下，选择“File”菜单，“Open”选项，打开刚才编辑好的文件 springimp.m，对此 M 文件作下述修改：

(1) 将第 5 行的 den=[1 5 20]改为 den=[1 5 20 0]（行号可参看 MATLAB Editor/Debugger 右下角的“Line”。

(2) 将第 27 行至第 29 行改为：

```
title('Spring System Impuls response');
xlabel('Time-sec');
ylabel('Displacement-meter');
```

选择“File”菜单的“Save as”选项，另存的文件名为“springstep.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 springstep，在 MATLAB Command Window 下查看运行的结果。

小结：本例从另一个角度再一次讨论了时间域分析方法和频率域分析方法之间的关系。利用部分分式展开和拉普拉斯逆变换的方法可以非常方便地将系统的频率域数据转化为时间域的数据。在展开的分式中，每一个分式就代表系统的一个运动模态。当然，在信号分析工具箱里 MATLAB 还提供了直接求解 s 函数拉普拉斯逆变换数值解的函数，但从概念上讲，还是用部分分式展开的方法更能反映控制系统的运动本质。本例编写的通过传递函数的部分分式展开求解系统时间响应的程序有一定的通用性，其核心代码能够求解不包含多重极点的各种传递函数的时间响应。读者可以将其改编为 M 函数，存在 MATLAB 的缺省路径下，方便以后的使用。程序中用到了 size ( ) 和 length ( )，这分别是用来求取矩阵和向量大小的函数。其含义和功能都非常简单，读者查阅 MATLAB 帮助就可以理解和使用，这里就不再赘述了。

residue ( ) 函数还可以根据部分分式展开求解原传递函数表达式，其调用格式为：

[B,A] = RESIDUE(R,P,K)

各参数的含义与上文的解释完全一致。例如对于前例的结果，在 MATLAB Command Window 下直接键入：

```
r =   0 - 0.1348i
      0 + 0.1348i
p = -2.5000 + 3.7081i
    -2.5000 - 3.7081I]
k =   []
[num,den]=residue(r,p,k)
num =   0      1
den =    1      5      20
```

可以看到，其结果与原传递函数完全相同。

相对于微分方程来说，控制系统的传递函数描述虽然是一种间接的方法，但其含义和数学基础同样是非常明确和坚实的。除了上文介绍的这些函数外，MATLAB 在复频域分析方面还有许多功能强大的函数，读者也可以编写 M 函数组建自己的工具箱，满足不同场合的特殊需要。

以上我们讨论的主要是 SISO 系统的数学描述，属于经典控制理论领域。下一节介绍的系统的状态方程描述是一种囊括了 SISO 系统和 MIMO 系统的数学描述方法，是所谓现代控制理论的基础。

8.3 控制系统的状态方程描述

列出控制系统的原始运动方程组后，固然可以如上节所说的那样化成关于单一变量的高阶微分方程，但也可以化成另一种标准形式的方程组，就是状态方程组。用状态方程组描述动态系统，是现代时间域控制理论学派（状态空间学派）的一种基本手段。在用状态方程描

述运动对象时，状态变量是最重要的基本概念之一。状态变量是这样来定义的：在描述对象运动的所有变量中，必定可以找到数目最少的一组变量，它们已经足以描述对象的全部运动。这组变量称为对象的状态变量。所谓足以描述系统的全部运动，是指：只要确定了这组变量在某一初始时刻 t=T 值，并且确定了从这一初始时刻起（t≥T）的输入量函数，则对象的全部变量在此刻和此后（t≥0）的运动都惟一确定了。

集总参数的线性网络可用微分方程表示为：

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad y(t) = Cx(t) + Du(t)$$

该系统的一阶微分方程即为状态方程，X 是状态变量。状态空间方法易于采用数字和模拟计算机求解，MATLAB 的控制系统工具箱中也提供了大量功能强大的函数可以用来进行状态方程的求解和仿真。另外，状态空间方法容易拓展到非线性系统。有两种方法可以得到系统的状态方程。一种是从系统的 n 阶微分方程得到；另一种是从系统模型中选用合适的状态变量直接写出。本节主要讨论第一种方法，如果不作特殊说明，均假设系统的微分方程已知。

状态向量是状态空间控制理论的基本概念。在状态空间控制理论中使用状态方程来描述动态系统的运动。状态方程的主要特征是：在全部受控量中，只选择一组状态变量来列写方程，其他受控变量不进入方程；状态方程必须写成标准形式。本节将详细讨论状态方程的数学描述，及其各种标准形式。

8.3.1　数学描述

假设一个 n 阶线性系统的微分方程描述如下所示。我们将讨论如何选取状态变量，得到该系统的状态方程描述，并给出状态方程在 MATLAB 环境下的实现方法。

$$a_n\frac{\mathrm{d}^n x}{\mathrm{d}t^n}+a_{n-1}\frac{\mathrm{d}^{n-1}x}{\mathrm{d}t^{n-1}}+\cdots+a_1\frac{\mathrm{d}x}{\mathrm{d}t}+a_0x$$

$$=b_n\frac{\mathrm{d}^n u}{\mathrm{d}t^n}+b_{n-1}\frac{\mathrm{d}^{n-1}u}{\mathrm{d}t^{n-1}}+\cdots+b_1\frac{\mathrm{d}u}{\mathrm{d}t}+b_0u$$

其中 $x(t),u(t)$ 分别是系统的输出量和输入量，$a_n \neq 0$。先计算列向量：

$$\begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \vdots\\ \beta_n\end{pmatrix}=\begin{pmatrix}a_n & & & & \\ a_{n-1} & a_n & & 0 & \\ a_{n-2} & a_{n-1} & a_n & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_0 & a_1 & a_2 & \cdots & a_n\end{pmatrix}^{-1}\begin{pmatrix}b_n\\ b_{n-1}\\ b_{n-2}\\ \vdots\\ b_0\end{pmatrix}$$

然后用下列方程组替代系统的原微分方程：

$$\begin{pmatrix}\dot{\xi}_1\\ \dot{\xi}_2\\ \vdots\\ \dot{\xi}_{n-1}\\ \dot{\xi}_n\end{pmatrix}=\begin{pmatrix} & \vdots & & \\ 0 & \vdots & I_{n-1} & \\ \cdots & \cdots & \cdots & \cdots\\ -\dfrac{a_0}{a_n} & -\dfrac{a_1}{a_n} & \cdots & -\dfrac{a_{n-1}}{a_n}\end{pmatrix}\begin{pmatrix}\xi_1\\ \xi_2\\ \vdots\\ \xi_{n-1}\\ \xi_n\end{pmatrix}+\begin{pmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_{n-1}\\ \beta_n\end{pmatrix}u$$

$$x=\xi_1+\beta_0u$$

这样就用 n 个一阶微分方程和一个代数方程代替了 n 阶的原始微分方程，与此同时引入了 n 个新的受控量，连同 x，一共是 $n+1$ 个受控量。新方程均不含输入量 u 的导数项。转化为状态方程的标准形式，可以得到：

$$A=\begin{pmatrix} & \vdots & & \\ 0 & \vdots & I_{n-1} & \\ \cdots & \cdots & \cdots & \cdots \\ -\dfrac{a_0}{a_n} & -\dfrac{a_1}{a_n} & \cdots & -\dfrac{a_{n-1}}{a_n} \end{pmatrix} \qquad B=\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{pmatrix}$$

$$C=(1 \quad 0 \quad \cdots \quad 0)_{1\times n} \qquad D=\beta_0$$

这是方程右边包含输入量导数项的情况。而不包含输入量导数项的情况则比较简单，只需选择输出量 x 的各阶导数作为 n 个状态变量即可，这里就不再赘述了。

与传递函数的表示方法类似，在 MATLAB 环境下可以很直观地将系统的状态方程模型表示为 4 个不同维数的矩阵（A，B，C，D），然后就可以使用 MATLAB 提供的控制系统工具箱中的各种函数对该系统操作了。请看下面这个著名的例子（Automatic Control Systems，作者 B. C. Kuo，第七版）。

如图 8-14 所示的电磁控制系统，磁性物质做成的小球在线圈磁力和自身重力的作用下产生运动。设小球质量 $M=0.05$ kg，磁力系数 $K=0.0001$，电感 $L=0.01$H，电阻 $R=1\,\Omega$，重力加速度 $g=9.81$ m/s^2；V 是电源电压，i 是线圈中电流，h 是小球与线圈的距离。根据系统的微分方程，试选取合适的状态变量，给出系统的状态方程，并讨论 h=0.01m，V=1V 时系统的运动情况。

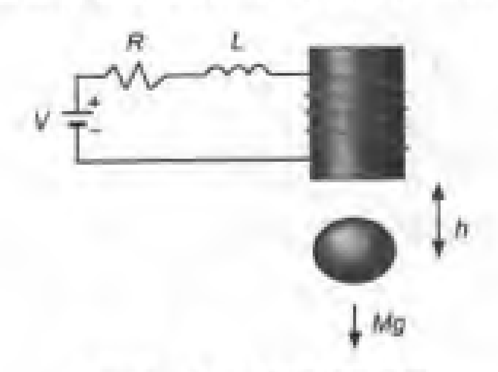


图 8-14　电磁控制系统示意图

结果：系统的微分方程为：

$$M\frac{\mathrm{d}^2 h}{\mathrm{d}t^2}+K\frac{i^2}{h}-Mg=0 \qquad L\frac{\mathrm{d}i}{\mathrm{d}t}+iR-V=0$$

选择系统的状态变量为 $X=\begin{pmatrix}\Delta h & \Delta\dot{h} & \Delta i\end{pmatrix}^T$，输出量为 y=h，输入量 u=V，可得系统的状态方程：

$$\dot{X}=\begin{pmatrix} 0 & 1 & 0 \\ 980 & 0 & -2.8 \\ 0 & 0 & 100 \end{pmatrix}X+\begin{pmatrix} 0 \\ 0 \\ 100 \end{pmatrix}u$$

$$y=(1 \quad 0 \quad 0)X$$

系统的开环时间响应如图 8-15 所示。

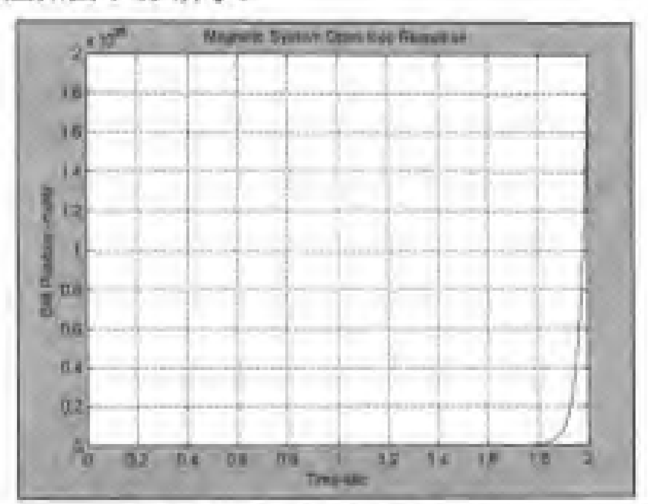


图 8-15　电磁控制系统开环响应

分析：本例系统的微分方程是不包含输入量导数项的情况，因而可以直接选取状态变量。本例选取的状态变量是距离 Δh、Δh 的一阶导数和电流 Δi，这是最一般的思路。因为所选的量都是相对变化量，所以加 Δ 号作标记。从图 8-15 系统的开环时间响应来看，相对距离 Δh 从零开始逐渐增大，直至 $+\infty$。这是因为系统的磁力不足以抵销小球的重力，小球在重力加速度的作用下逐渐远离带电线圈的缘故。根据图 8-15 的信息，我们可以判定系统的开环响应是不稳定的。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "magneticsys.m"：

```
%关闭所有图形窗口
close all;
%状态方程模型
A=[0 1 0;980 0 -2.8;0 0 -100];
B=[0;0;100];
C=[1 0 0];
D=0;
%仿真时间初始化
t = 0:0.01:0.2;
u = zeros(size(t))+1;
x0 = [0.01 0 0];
[y,x] = lsim(A,B,C,0,u,t,x0);
h = x(:,1);
%绘图
plot(t,h)
title('Magnetic System Open-loop Response');
xlabel('Time-sec');
```

```
ylabel('Ball Position -meter');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“magneticsys.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 magneticsys，在 MATLAB Command Window 下查看运行的结果。

小结：本例简要介绍了选择状态变量和求取系统状态方程的方法，并给出了状态方程在 MATLAB 环境下的表示方法和处理手段。程序中用到了 lsim ( ) 函数，其功能是仿真 LTI 系统强迫输入的时间响应。该函数的基本调用格式为：

```
[Y,X] = LSIM(SYS,U,T,X0...)
```

其中 SYS 就是系统的状态方程参数（A，B，C，D)，U 是输入量，T 是仿真时间向量，X0 是系统初值，Y 是返回的输出量，X 是返回的状态变量的历史数据。

对于离散控制系统来说，也可以用状态方程对其进行数学描述。将其差分方程化为状态方程模型后，有：

$$\begin{cases} x(k+1) = Gx(k) + Hu(k) \\ y(k) = Cx(k) + Du(k) \end{cases}$$

简记为（G，H，C，D)。在 MATLAB 环境下，其表示方法和处理手段与连续 LTI 系统并无区别。事实上，使用数字式计算机进行仿真时，连续系统也要先进行采样量化，化为离散系统，然后再进行处理。因此，从实际操作的角度来看，离散控制系统和连续系统并无多大区别。当然，在进行理论研究时，还是要注意将二者区分开来。

8.3.2 对角化与 Jordan 标准型

从本节开始，我们将讨论系统的状态方程模型的各种标准型。

给定 LTI 系统（A，B，C，D)，输入量 u（t）和状态变量的初值 x_0 后，可以求出其状态方程的解为：

$$x(t) = e^{At} x_0 + \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau$$

要求解此方程，很自然就想到将矩阵 A 对角化，将其化为对角元为特征值的对角矩阵。这样一来，上述方程里的矩阵指数也就化成了对角元是 $e^{\lambda_1 t}, e^{\lambda_2 t}, \mathrm{L}, e^{\lambda_n t}$ 的对角矩阵（假定 A 有 n 个互不相同的特征值）。

对于线性系统（A，B，C，D）来说，我们可以找到非奇异变换矩阵 W，使得：

$$A' = W^{-1}AW = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

$$B' = W^{-1}B \qquad C' = CW \qquad D' = D$$

此系统的新的表示形式（A'，B'，C'，D'）称为系统的对角规范型。W 是矩阵 A 的特征向量组成的矩阵。对应的状态变量变为 X＝WX'。请看下例：

给定某控制系统的状态空间描述为：



$$\dot{X}=\begin{pmatrix}0 & 1 & -1\\ -6 & -11 & 6\\ -6 & -11 & 5\end{pmatrix}X+\begin{pmatrix}0\\0\\1\end{pmatrix}u$$

$$y=\begin{pmatrix}1 & 0 & 0\end{pmatrix}X$$

试求其对角规范型和变换矩阵 W，并根据其对角规范型绘制系统在初值为 X=[5;10;15]，u=0 时的响应曲线。

结果：系统的对角规范型为

The Diagonal Form of System is:

```
L = -1.0000    -0.0000    -0.0000
     0.0000    -2.0000    -0.0000
    -0.0000     0.0000    -3.0000
b = -2.8284
    -13.7477
    10.8628
c = 0.7071    -0.2182    -0.0921
```

系统对角规范型的变换矩阵为

Transformation Matrix is:

```
W = 0.7071    -0.2182    -0.0921
    0.0000    -0.4364    -0.5523
    0.7071    -0.8729    -0.8285
```

系统在 u=0 时的响应曲线如图 8-16 所示。

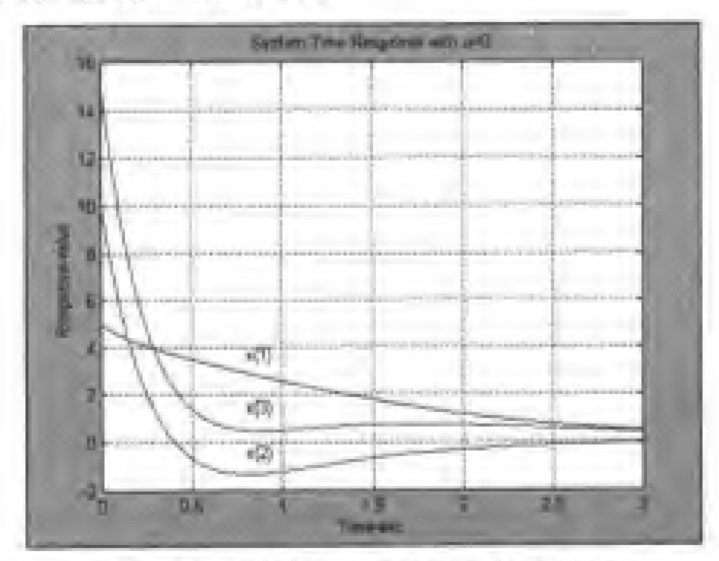


图 8-16　系统状态方程模型时间响应曲线

分析：从系统的响应曲线来看，各状态变量最终都趋于零。这一方面是因为输入量 u=0，另一方面是因为初值不为零，但矩阵 A 的特征值均为负值，系统的零输入响应呈衰减趋势。但状态变量 x(3)下降幅度和速度最大，x(1)下降幅度和速度最小。这是因为状态变量 x(3)对应的特征值是-3，而 x(1)对应的特征值是-1，其衰减速率不同。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 “diagform.m”：

```
%关闭所有图形窗口
close all;
%状态方程模型
A=[0 1 -1; -6 -11 6; -6 -11 5];
B=[0;0;1];
C=[1 0 0];
%求取对角规范型
[W,lamda]=eig(A);
L=inv(W)*A*W;
b=inv(W)*B;
c=C*W;
%显示结果
disp('The Diagonal Canonical Form of System is:');
L
b
c
disp('Transformation Matrix is:');
W
%仿真数据初始化
t=0:0.01:3;
x0=[5;10;15];
xx0=inv(W)*x0;
n=length(t);
x=zeros(3,n);
xx=zeros(3,n);
%求解状态变量
for i=1:n
xx(:,i)=expm(L*t(i))*xx0;
x(:,i)=W*xx(:,i);
end
%绘图
plot(t,x)
title('System Time Response with u=0');
xlabel('Time-sec');
ylabel('Response-value');
text(0.8,3.7,'x(1)');
```

```
text(0.8,1.5,'x(3)');
text(0.8,-0.4,'x(2)');
grid;
```

选择"File"菜单的"Save"选项，保存文件名为"diagform.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 diagform，在 MATLAB Command Window 下查看运行的结果。

小结：通过对角规范型求解系统状态方程的解的优点是，不必计算矩阵指数，只需计算单个的标量的指数即可。这样的好处是可以避免计算矩阵指数带来的算法的不稳定性。当然，由于不必手工计算，在 MATLAB 环境下这两种算法在程序编制的工作量上没有什么太大的差别，但在计算复杂性上还是有一定差别的。在本例具体求解的过程中，需要注意的是求得对角规范型的状态变量 xx 的时间响应之后，还要用变换矩阵 W 将其转化为原状态变量 x，然后再绘图。另外，如果变换矩阵 W 的向量排列的顺序不同，所求得的对角规范型在排列顺序上也稍有差别。

熟悉线性代数的读者都清楚，如果 n×n 矩阵 A 的线性无关特征向量个数小于 n 时，就无法将其对角化，也就无法将该线性系统化为对角规范型。在这种情况下，只能将其化为约当规范型（Jordan Canonical Form）。

$$A' = V^{-1}AV = J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{pmatrix}$$

设 A 矩阵有重特征值，其相异的特征值为 p 个。则必存在非奇异线性变换矩阵 V，将其化为约当型

其中
$$J_i = \begin{pmatrix} \lambda_i & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \lambda_i & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \lambda_i \end{pmatrix}$$

变换矩阵 V 是由 A 矩阵对应特征值的广义特征向量组成的。即

$$V = \left(V_1 \ \vdots \ V_2 \ \vdots \ \cdots \ \vdots \ V_p\right)$$

$$AV_i = V_i J_i, \qquad (i = 1, 2, \cdots, p)$$

与对角规范型相似，B'矩阵和 C'矩阵也可通过变换矩阵 V 得到，只需将表达式中的 W 换成 V 即可。MATLAB 控制系统工具箱提供了一个求取矩阵 A 规范型的函数 jordan ()，其基本调用格式为：

[V,J] = JORDAN(A)

其中 V 是变换矩阵，其列向量是矩阵 A 的广义特征向量。J 矩阵是求得的约当规范型。其具体用法请看下面这个简单的例子。

给定某控制系统的状态方程模型如下，试求其约当规范型。

$$\dot{X}=\begin{pmatrix}4 & 6 & 0\\ -3 & -5 & 0\\ -3 & 6 & 1\end{pmatrix}X+\begin{pmatrix}1\\ 2\\ 3\end{pmatrix}u$$

$$y=\begin{pmatrix}1 & 0 & 0\end{pmatrix}X$$

结果：系统的约当规范型及其变换矩阵为

```
The Jordan Canonical Form of System is
J =   -2      0      0
       0      1      1
       0      0      1
b =   5.0000
     -0.7500
      3.0000
c =   -1      0      2
The Transformation Matrix is
V =   -1      0      2
       1      0     -1
      -3    -12      3
```

分析：从结果来看，系统矩阵 A 的约当标准型有两个约当块。第一个约当块阶次为 1，对应变换矩阵 V 中第一列列向量；第二个约当块阶次为 2，对应变换矩阵 V 中第二和第三列列向量，这两列向量也称为矩阵 A 的广义特征向量。根据前边介绍的 jordan（）函数，可以求出这些值。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "jordanform.m"：

```
%关闭所有图形窗口
close all;
%状态方程模型
A=[4 6 0;-3 -5 0;-3 6 1];
B=[1;2;3];
C=[1 0 0];
D=0;
%求取系统的约当规范型
[V,J]=jordan(A);
b=inv(V)*B;
c=C*V;
%显示结果
disp('The Jordan Canonical Form of System are');
J
```

```
b
c
disp('The Transformation Matrix is');
V
```

选择“File”菜单的“Save”选项，保存文件名为“jordanform.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 jordanform，在 MATLAB Command Window 下查看运行的结果。

小结：jordan ( ) 函数对输入矩阵 A 的限制比较严格，要求矩阵 A 的元素必须是整数或绝对值较小的整数之比。如果不满足这个条件的话，求得的约当规范型将与其真实值相差很远。另外，MATLAB 还提供了一条函数 canon ( )，也可以用来求解系统的约当规范型和伴随规范型，其基本调用格式为：

[CSYS,T] = CANON(SYS,TYPE)

其中 SYS 表示系统的原状态方程参数（A，B，C，D）；TYPE 表示希望求得规范型的类型，'modal' 代表约当规范型，'companion'代表伴随规范型；CSYS 表示求得的规范型参数（A'，B'，C'，D'）；T 表示变换矩阵。

但这条函数有一个缺憾，就是 A 矩阵在无法完全对角化时有可能得出错误的结果。虽然此时 MATLAB 会给出警告，但我们还是建议在求解系统的约当规范型时，使用 jordan ( ) 函数。

8.3.3　可控规范型

状态可控性的含义是系统控制输入 u（t）支配状态变量 X（t）的能力。简单来说，就是系统的状态变量 X（t）能否在输入量 u（t）的控制下，从 n 维空间的任意一点出发，达到指定的 n 维空间的某一点。

某一状态变量 x（t）满足上述要求，则称该状态变量是可控的。若全体状态变量 X（t）均满足要求，则称该系统是完全可控的。

通过简单的推导，可得出线性定常系统状态可控性的代数判据为：

线性定常系统（A，B，C，D）状态完全可控的充分必要条件是，系统的可控性矩阵的秩为 n。

$$Q_c = \left(B \ \vdots \ AB \ \vdots \ \cdots \ \vdots \ A^{n-1}B\right)$$

如果系统是状态完全可控的，则可以从可控性矩阵中挑出 n 个线性无关的列向量，以它们或它们的线性组合作为变换矩阵，就可以导出系统的第一可控规范型。系统的第一可控规范型为：

$$A_{c1} = \begin{pmatrix} & & & & \vdots & -a_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & & & & \vdots & -a_{n-1} \\ & 1 & & & \vdots & -a_{n-2} \\ & & \ddots & & \vdots & \vdots \\ & & & 1 & \vdots & -a_1 \end{pmatrix} \qquad B_{c1} = \begin{pmatrix} 1 \\ \cdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$C_{c1} = (CB \ \vdots \ CAB \ \vdots \ \cdots \ \vdots \ CA^{n-1}B)$$

其中：
$$\begin{cases} B=Q_c B_{c1} \\ AQ_c=Q_c A_{c1} \\ CQ_c=C_{c1} \end{cases}$$

这里假设 B 是 n 维列向量，则取变换矩阵就是系统可控性矩阵本身。此单输入系统的第一可控规范型的方框图如图 8-17 所示。

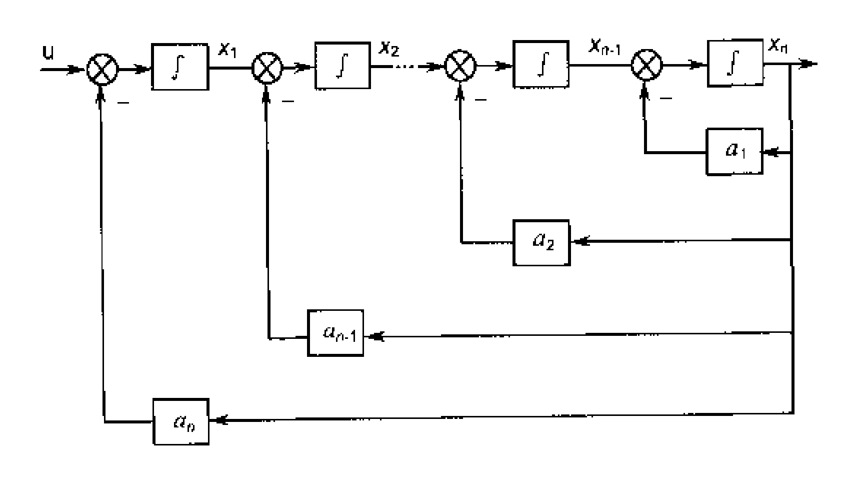


图 8-17　系统第一可控规范型方框图

图中的主通道是积分器串联形式，状态变量是每个积分器的输出，反馈通道是第一可控规范型矩阵 A 的最后一列向量。从图中可以看出，控制输入量 u 位于积分器串联链的最前方，因此，在输入端 u 加控制信号，可以控制系统的所有状态变量。这也是系统可控性的基本含义。

除系统的第一可控规范型之外，控制系统中常用的还有第二可控规范型等其他一些可控规范型。

这些规范型的形式和求解方法与第一可控规范型大同小异，都是首先根据可控性矩阵的某种线性变换来求解其变换矩阵，然后再求解可控规范型本身。因此，在以下的讨论中，我们一般只考虑第一可控规范型的情况。如果不特别指明，本书中的可控规范型都是指系统的第一可控规范型。

虽然 MATLAB 环境没有直接提供求解系统可控规范型的函数，但依靠本书第 1 章介绍过的有关矩阵运算的函数，可以很轻松地实现原系统与其可控规范型之间的相互转换。请看下面这个简单的例子：

给定线性定常系统状态方程模型如下，试判断系统的可控性。如果系统状态完全可控，试求其可控规范型和变换矩阵。

$$A=\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 4 & 1 & 3 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \qquad B=\begin{pmatrix} 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$
$$C=(1 \quad 1 \quad 0 \quad 0)$$

结果：系统完全可控，其可控标准型为

```
System is   Controllable.
System First Controllable Canonnical Form is:
Ac1 =     0          0          0          -64.0000
          1.0000     0          -0.0000     96.0000
          -0.0000    1.0000     0          -52.0000
          0.0000     0          1.0000      12.0000
Bc1 =1
     0
     0
     0
Cc1 =1     11     58     268
The Transformation Matrix is:
Q =  1     4     12     32
     0     7     46     236
     1     6     28     120
     2     4      8      16
```

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "controlform.m"：

```
%系统状态方程模型
A=[2 0 0 1;0 4 1 3;0 0 4 1;0 0 0 2];
B=[1;0;1;2];
C=[1 1 0 0];
%系统阶次
n=length(A);
%求解系统可控性矩阵
Q=zeros(n);
Q(:,1)=B;
for i=2:n
%系统可控性矩阵的列向量
    Q(:,i)=A*Q(:,i-1);
end
%系统可控性矩阵的秩
m=rank(Q);
%判断系统是否状态完全可控，并求解可控规范型
if m==n
%系统状态完全可控
    Ac1=inv(Q)*A*Q;
    Bc1=inv(Q)*B;
```

```
    Cc1=C*Q;
%显示结果
    disp('System is    Controllable.');
    disp('System First Controllable Canonnical Form is:');
    Ac1
    Bc1
    Cc1
    disp('The Transformation Matrix is:');
    Q
else
%系统状态不完全可控
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
%可控的状态变量数
    m
end
```

选择“File”菜单的“Save”选项，保存文件名为“controlform.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 controlform，在 MATLAB Command Window 下查看运行的结果。

小结：本例的核心代码可以作为一个求解系统第一可控规范型的 M 函数单独使用。当然，这个算法仅仅考虑了最简单的情况，并且没有编写注释。程序中使用了 rank（ ）函数来判断系统可控性矩阵 Q 是否满秩，这牵扯到一系列病态条件讨论和算法稳定性的问题，这里就不再介绍了，感兴趣的读者请参阅相关的线性代数文献。

8.3.4 可观规范型

状态可观性的含义是系统输出量 y（t）反映状态变量 X（t）的能力。它回答了当系统输入量 u（t）=0 时能否通过输出量 y（t）的量测值确定状态变量 X（t）的问题。想要精确定义系统的完全可观，首先要定义所谓系统的不可观测状态：在有限的时间区间[T，T']内，把状态空间某个非零的有限点 x'作为系统的初态，x（T）=x'，产生的时间响应在该时间区间内恒有 y（t）≡0，则称状态 x'为系统在时间区间[T，T']上的不可观测状态。如果状态空间中不存在不可观测状态，则称系统是在[T，T']上状态完全可观测的。

与系统的可控性相似，通过简单的推导，可得出线性定常系统状态可控性的代数判据为：线性定常系统（A，B，C，D）状态完全可观的充分必要条件是，系统的可观性矩阵的秩为 n。

$$Q_o=\begin{pmatrix}C & \vdots & CA & \vdots & \cdots & \vdots & CA^{n-1}\end{pmatrix}^T$$

如果系统是状态完全可观的，则可以从可观性矩阵中挑出 n 个线性无关的列向量，以它们或它们的线性组合作为变换矩阵的逆，就可以导出系统的第一可观规范型。系统的第一可观规范型为

$$A_{o1}=\begin{pmatrix} & \vdots & & I_{n-1} & \\ & \vdots & & & \\ & \vdots & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -a_n & \vdots & -a_{n-1} & \cdots & -a_1 \end{pmatrix} \quad B_{o1}=\begin{pmatrix} CB \\ \cdots \\ \vdots \\ \cdots \\ CA^{n-1}B \end{pmatrix}$$

$$C=(1 \ \vdots \ 0 \ \cdots \ 0)$$

其中变换矩阵：

$$P^{-1}=Q_o=[C \ \vdots \ CA \ \vdots \ \cdots \ \vdots \ CA^{n-1}]^T, \quad \begin{cases} B=PB_{o1} \\ AP=PA_{o1} \\ CP=C_{o1} \end{cases}$$

状态完全可观系统的方框图与状态完全可控系统的方框图基本一致，也是由积分器串联链构成。只是串联链的最后变为系统的输出量 y（t），从而通过 y（t）可以观测所有的状态变量 X（t）。与系统可控性的情况类似，除第一可观规范型外还有第二和其他的一些可观规范型，这里就不再赘述了。同样，如果不特别指明的话，以下的系统可观规范型都是指系统的第一可观规范型。

对于上例的线性定常系统，试判断系统的可观性。如果系统状态完全可观，试求其可观规范型和变换矩阵。

结果：系统完全可观，其可观规范型和变换矩阵为

$$A=\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 4 & 1 & 3 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \quad B=\begin{pmatrix} 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

$$C=(1 \ \ 1 \ \ 0 \ \ 0)$$

```
System is   Observable.
System First Observable Canonnical Form is:
Ao1 =-0.0000      1.0000        0          0
          0          0       1.0000        0
       0.0000     -0.0000       0        1.0000
     -64.0000    96.0000    -52.0000    12.0000
Bo1 =1.0000
     11.0000
     58.0000
     268.0000
Co1 = 1       0       0       0
The Transformation Matrix is:
P = -14.0000    16.0000    -5.3750     0.5625
```

```
15.0000    -16.0000    5.3750    -0.5625
 0          1.0000    -0.7500     0.1250
-8.0000     8.0000    -2.5000     0.2500
```

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "observeform.m":

```
%系统状态方程模型
A=[2 0 0 1;0 4 1 3;0 0 4 1;0 0 0 2];
B=[1;0;1;2];
C=[1 1 0 0];
%系统阶次
n=length(A);
%求解系统可观性矩阵
Q=zeros(n);
Q(1,:)=C;
for i=2:n
%系统可观性矩阵的列向量
    Q(i,:)=Q(i-1,:)*A;
End
%系统可观性矩阵的秩
m=rank(Q);
%判断系统是否状态完全可观，并求解可观规范型
if m==n
%系统完全可观
    P=inv(Q);
    Ao1=inv(P)*A*P;
    Bo1=inv(P)*B;
    Co1=C*P;
%显示结果
    disp('System is    Observable.');
    disp('System First Observable Canonnical Form is:');
    Ao1
    Bo1
    Co1
    disp('The Transformation Matrix is:');
%状态变换矩阵
    P
else
%系统状态不完全可观
```

```
    disp('System State Variables cannot be totally Observed');
    disp('The rank of System Observable Matrix is:');
%可观的状态变量数
    m
end
```

选择“File”菜单的“Save”选项，保存文件名为“observeform.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 observeform，在 MATLAB Command Window 下查看运行的结果。

小结：对照前例的系统可控规范型可以看到，可观规范型矩阵 Ao1 最后一行与可控规范型矩阵 Ac1 最后一列的元素完全相同，都是[-64.0000　96.0000　-52.0000　12.0000]。事实上，在以后有关控制系统各数学描述间相互转换的章节中就会了解，这个行向量的元素就是该系统传递函数分母多项式的系数，也称为矩阵 A 的特征多项式的系数。

有关系统状态空间规范型的介绍就到此为止，研究规范型对系统的分析和综合都有十分重要的意义。规范型可以把系统的某些特性表现得更充分、更明显，规范型的各系数矩阵的元素往往有十分简洁的形式，给系统的分析和综合带来极大的方便。

采用状态空间法综合和设计系统时，常常采用非奇异线性变换将系统化为特定的规范型式。再对规范型进行综合和设计，这样的综合或设计方法是规范化的，便于使用计算机辅助计算。

从以上的几个实例中我们也可以看到，使用 MATLAB 求解系统的规范型是非常简洁并且有效的。有了 MATLAB 这个得力的工具，我们还可以根据不同的实际需要，自己定义系统规范型系数矩阵的特定结构，并构造适当的非奇异线性变换矩阵，将系统化为该规范型。用 MATLAB 探索和实现这些想法都是非常方便的。

8.4　控制系统模型转换

本章前三节主要介绍了控制系统的微分方程描述、传递函数描述和状态方程描述。其中微分方程描述是整个控制系统数学描述的基础，传递函数描述和状态方程描述都是在微分方程描述的基础上发展起来的。但由于形式不够简洁，处理和运算不够方便，一般在控制系统仿真和设计时已经很少直接用到微分方程描述了。但因为人们习惯使用的传递函数描述和状态方程描述分属于频率域和时间域，这样一来就存在一个模型间相互转换的问题。当然，正如前文所述，我们可以先把传递函数描述或状态方程描述化为微分方程描述，这只需将前几节的有关推导作逆运算即可，然后再化成希望的描述形式。不过 MATLAB 提供了直接进行控制系统模型间相互转换的函数，可以完成传递函数描述与状态方程描述之间的相互转换，控制系统方框图与传递函数描述和状态方程描述之间的转换，以及状态方程描述的最小实现。

8.4.1　传递函数向状态方程的转换

在实际的控制系统仿真和设计中，系统的结构和参数基本上是未知的。这时要想通过分

析的方法建立系统的状态方程描述是非常困难的，甚至是不可能的。一个可行的办法是，先用实验的方法确定其输入与输出之间的关系，比如说确定它的传递函数。然后，根据系统的传递函数再确定系统的状态方程和输出方程。由系统的传递函数或脉冲响应函数来建立与其输入输出特性上等价的状态方程描述称为实现问题。所找到的状态方程，称为该传递函数的一个实现。实现问题是控制理论和控制工程的一个基本问题。

给定系统的传递函数 G（s），可以找到各种各样结构的实现（A，B，C，D）。如果找到的实现对矩阵（A，B）是完全可控的，就称为可控性实现；如果对矩阵（A，C）是完全可观的，就称为可观性实现；如果矩阵 A 是约当规范型，就称为约当型实现。在各种各样的实现中，最感兴趣的是矩阵 A 的阶次最低的实现，称之为最小实现。显而易见，最小实现的结构最简单，按最小实现模拟原系统是最方便、最经济的。通过理论推导我们可以得到下边的结论：

（A，B，C，D）是传递函数 G（s）的最小实现的充分必要条件是，（A，B，C，D）是完全可控又可观的。

G（s）的最小实现也不是惟一的，它们维数相同，系数矩阵 A 之间是非奇异线性变换的关系。

MATLAB 提供了一条函数 tf2ss ( )，可以根据系统的传递函数描述求解其状态方程描述。其基本调用格式为：

[A,B,C,D] = TF2SS(NUM,DEN)

其中 NUM 是传递函数分子多项式系数的降幂排列，DEN 是传递函数分母多项式系数的降幂排列；（A，B，C，D）就是实现该传递函数的系统状态方程描述，这个实现是系统的可控性实现。

MATLAB 还提供了一条函数 minreal ( )，可以求解系统状态方程的最小实现。不过这条函数不是根据传递函数来求解其最小实现，而是根据系统的状态方程描述来求解其最小实现。其调用格式为：

SYSr = MINREAL(SYS)

其中 SYS 是原系统的状态方程描述（A，B，C，D），SYSr 是系统的最小实现（As，Bs，Cs，Ds）。请看下面这个简单的例子：

某控制系统的传递函数如下，试求其可控性实现。并判断该实现是否是系统的最小实现。如果不是，则给出系统的一个最小实现。

$$G(s) = \frac{1}{s^2(s^2 - s + 1)}\begin{pmatrix} s^2 - s \\ s^2 \end{pmatrix}$$

结果：系统的可控性实现为

system Controller Realization is:

```
A =  -1    1    0    0
      1    0    0    0
      0    1    0    0
      0    0    1    0
B =   1
```

```
     0
     0
     0
C =  0     1    -1    0
     0     1     0    0
D =  0
     0
```

该实现不是系统的最小实现，最小实现为

```
1 state(s) removed
system Minimal Realization is:
As = -1.0000    0.5257    0.8507
      0.5257    0.4472    0.7236
      0.8507   -0.2764   -0.4472
Bs = 1
     0
     0
Cs = 0   -0.3249    1.3764
     0    0.5257    0.8507
Ds = 0
     0
```

分析：本例给出的系统传递函数确切的说应该称作传递函数矩阵。当系统的输入量或输出量不只一个时（也就是 MIMO 系统），就需要用传递函数矩阵来表示系统的输入量与输出量之间的关系。本例 2×1 的传递函数矩阵表示有两个输出量，一个输入量。从结果来看，无论是系统的可控性实现还是最小实现，与输出量相关联的矩阵 C 都是由二维行向量构成，而与输入量相关联的矩阵 B 都是一维的列向量。

本例要求判断系统的可控性实现是否是最小实现，可以通过判断该实现的可观性来解决。如果系统完全可观，那么该可控性实现就是系统的最小实现；否则就根据该可控性实现调用 minreal（）函数求解系统的最小实现。

从结果来看，通过 tf2ss（）函数求出的系统可控性实现的 A 矩阵和 C 矩阵的最后一列的元素均为零，这说明最后一个状态变量在状态方程中没有反映，无法通过输出量 Y（二维）来观测。minreal（）函数也给出结果“1 state(s) removed”，消去了一个状态变量。也就是说，原系统四阶的传递函数通过三维的状态方程系数矩阵就可以实现了，系统的传递函数中存在零极相消。

求解过程：在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件“realization.m”：

```
%系统的传递函数描述
num=[1 -1 0;1 0 0];
den=[1 1 -1 0 0];
%求解系统的可控性实现
```

```
[A,B,C,D]=tf2ss(num,den);
disp('system Controller Realization is:');
A
B
C
D
%求解可观性矩阵
n=length(A);
m=size(C,1);
%系统可观性矩阵
Q=zeros(m*n,n);
for i=1:m
    Q(i,:)=C(i,:);
end
for i=1:n-1
    for j=1:m
        Q(i*m+j,:)=Q((i-1)*m+j,:)*A;
    end
end
%判断是否是最小实现
r=rank(Q);
%是最小实现
if r==n
    disp('Original (A,B,C,D) is system Minimal Realization');
%不是最小实现
else
    %求解系统的最小实现
    [As,Bs,Cs,Ds]=minreal(A,B,C,D);
    %显示结果
    disp('Original (A,B,C,D) is system Minimal Realization');
    disp('system Minimal Realization is:');
    As
    Bs
    Cs
    Ds
end
```

选择“File”菜单的“Save”选项，保存文件名为“realization.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 realization，在 MATLAB Command Window 下查看运行的结果。

小结：本例在判断系统的可观性时，使用的是改进了的前例的代码。前例的代码只能判断 SISO 系统的可观性，而本例的代码能够判断任意线性定常系统的可观性。读者也可以自己编制有关系统最小实现的 M 函数，这要用到马尔可夫（Markov）参数矩阵的一些知识，这里就不介绍了。

8.4.2　状态方程向传递函数的转换

给定线性定常系统（A，B，C，D），其状态方程和输出方程为

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

方程两边同时取拉普拉斯变换并代入消元，得

$$Y(s) = C(sI - A)^{-1}BU(s) + DU(s)$$

则系统的传递函数为：

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D$$

其中 $(sI - A)^{-1}$ 称为系统的预解矩阵。

可见，将系统的状态方程描述转换为传递函数描述的关键就在于预解矩阵的求解。我们在介绍矩阵指数时提到过，预解矩阵的拉普拉斯逆变换就是矩阵指数，而矩阵指数是求状态方程解的关键所在。因此，预解矩阵的分析和求解在系统的状态方程描述中具有非常重要的意义。

在手工计算时通常先按照法捷耶娃法来求解系统的预解矩阵，然后再根据状态方程系数矩阵求解系统的传递函数。MATLAB 提供了一条函数 ss2tf（）可以直接将系统的状态方程描述转换为传递函数描述，其基本调用格式为

[NUM,DEN] = SS2TF(A,B,C,D,iu)

其中（A，B，C，D）是系统的状态方程描述的系数矩阵，iu 表示对系统的第 iu 个输入量求传递函数；NUM 是返回的系统传递函数的分子多项式系数的降幂排列，如果是 MIMO 系统 NUM 就是矩阵，其行数与输出量 y 的个数一样多；DEN 是返回的系统传递函数的分母多项式系数的降幂排列，无论是 SISO 系统还是 MIMO 系统，DEN 都是行向量。请看下面这个简单的例子：

给定某控制系统的状态方程描述如下。试分别求其对第一和第二个输入的传递函数和零极点形式的传递函数。

结果：系统的传递函数和零极点模型为

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -2 \\ -22 & -11 & -4 & 0 \\ -23 & -6 & 0 & -6 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 3 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

```
num1 =    0    1.0000    4.0000   -0.0000   -0.0000
          0   -0.0000   -0.0000   -0.0000  -11.0000
den1 = 1.0000  10.0000  35.0000   50.0000   24.0000
Its zero-pole form is:
z1 =      0    Inf
          0    Inf
         -4    Inf
p1 = -4.0000
     -1.0000
     -2.0000
     -3.0000
k1 = 1.0000
   -11.0000
```

系统第二个输入函数为

```
num2 =    0    3.0000   12.0000   -0.0000   -0.0000
          0    1.0000    6.0000   11.0000  -27.0000
den2 = 1.0000  10.0000  35.0000   50.0000   24.0000
Its zero-pole form is:
z2 = -0.0000           -3.6557 + 2.6878i
      0.0000           -3.6557 - 2.6878i
     -4.0000            1.3114
p2 =  -4.0000
      -1.0000
      -2.0000
      -3.0000
k2 =   3
       1
```

传递函数 $G(s)=\dfrac{1}{s^4+10s^3+35s^2+50s+24}\begin{pmatrix} s^3 & 3s^3+12s^2 \\ -11 & s^3+6s^2+11s-27 \end{pmatrix}$

对应的零极点模型为

$$\frac{1}{(s+4)(s+1)(s+2)(s+3)}\begin{pmatrix} s^2(s+4) & 3s^2(s+4) \\ -11 & (s+3.66-2.69i)(s+3.66+2.69i)(s-1.31) \end{pmatrix}$$

分析：可以看出，此系统有两个输入量，两个输出量。因此，系统的传递函数为 2×2 的矩阵。结果中系统对第一个输入量传递函数的零点为“inf”，这表示该项是常数项，不包含 s 的幂。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件“transfunc.m”：

```
%系统的状态方程描述
A=[0 0 0 1;1 0 0 -2; -22 -11 -4 0; -23 -6 0 -6];
B=[0 0;0 0;0 1;1 3];
C=[0 0 0 1;0 0 1 0];
D=[0 0;0 0];
%第一个输入量的传递函数
[num1,den1]=ss2tf(A,B,C,D,1);
[z1,p1,k1]=ss2zp(A,B,C,D,1);
%第二个输入量的传递函数
[num2,den2]=ss2tf(A,B,C,D,2);
[z2,p2,k2]=ss2zp(A,B,C,D,2);
%显示结果
disp('System Transfer Function of the first input is:');
num1
den1
disp('Its zero-pole form is:');
z1
p1
k1
disp('System Transfer Function of the second input is:');
num2
den2
disp('Its zero-pole form is:');
z2
p2
k2
```

选择“File”菜单的“Save”选项，保存文件名为“transfunc.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 transfunc，在 MATLAB Command Window 下查看运行的结果。

小结：本例在求解系统的零点、极点和增益时，没有使用前边介绍过的函数 tf2zp ( )，而是使用了一个新的函数 ss2zp ( )。该函数的功能是直接根据系统的状态方程描述求取系统的零点、极点和增益，不必先化成传递函数描述形式。其基本调用格式与 tf2zp ( ) 函数类似，只是多了一个输入量选择的参数:

[Z,P,K]= SS2TF(A,B,C,D,iu)

其中（A，B，C，D）是系统的状态方程描述的系数矩阵；iu 表示对系统的第 iu 个输入量求零点、极点和增益；Z 是求得的零点矩阵；P 是求得的极点矩阵，Z 和 P 的列数与输出量 y 的个数一样多，行数视零点个数的多少而定；K 是求得的增益列向量，其维数等于输出量 y 的个数。

8.4.3 由方框图求状态方程和传递函数

在研究复杂的动态系统时，一般习惯利用框图来辅助列写系统的微分方程或传递函数。这样做的好处：一方面是直观，符合控制系统的实际情况；另一方面也方便系统模型的化简和运算，包括系统的综合和校正等等。系统方框图最基本的单元如图 8-18 所示，其闭环传递函数为

$$H(s)=\frac{1}{1+G(s)}$$

复杂系统的框图可能有许多个框，框与框之间的连接关系也错综复杂。这就给手工计算系统的传递函数和状态方程带来了很大的困难。考虑到计算的烦琐性，而数字式计算机在这方面的优势非常明显，因此，这些工作目前一般都由计算机辅助设计软件来完成。MATLAB 提供了一条函数 connect（）可以将方框图描述的线性系统转换为状态方程的描述形式。其基本调用格式为

SYSc = CONNECT(SYS,Q,INPUTS,OUTPUTS)

相关参数和返回值的说明请看下面例子的小结。

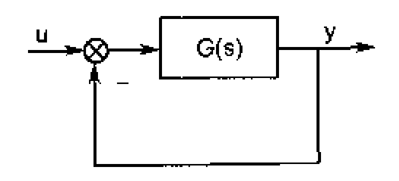


图 8-18 控制系统方框图的基本单元

某控制系统的方框图如图 8-19 所示。试求解其状态方程描述（A，B，C，D）和系统的传递函数。

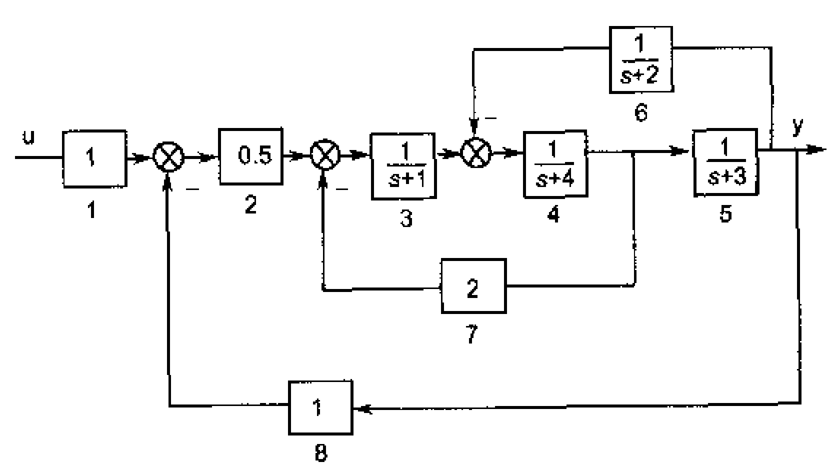


图 8-19 线性系统方框图

结果：系统的状态方程描述为

A = −1.0000	−1.0000	−0.5000	0
1.0000	−4.0000	0	−1.0000
0	1.0000	−3.0000	0

```
        0         0      1.0000    -2.0000
B = 0.5000
       0
       0
       0
C = 0     0     1     0
D = 0
```

系统的传递函数描述为

```
Transfer Function of the block-diagram is:
num =     0     0.0000    0.0000    0.5000    1.0000
den =  1.0000  10.0000   36.0000   56.5000   32.0000
```

对应传递函数为　　$G(s)=\dfrac{0.5s+1}{s^4+10s^3+36s^2+56.5s+32}$

分析：对于复杂系统框图仿真的问题，首先要将原系统方框图分解为单个的基本框图，并求解其关联矩阵；然后再寻找合适的 MATLAB 函数进行仿真。

本例的解题过程分为两部分：

1. 求解系统的关联矩阵

系统的关联矩阵 Q 是描述各框图间连接关系的矩阵。其每行第一个元素是框号，其余的元素依次是与该框连接的框号。图 8-19 各框的编号已经给出，例如第 3 框，它与第 2 框正连接，与第 7 框负连接，与其他框不相连，那么关联矩阵 Q 的第三行就是[3 2 -7]。依次类推，得到系统的关联矩阵为

$$Q=\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & -8 \\ 3 & 2 & -7 \\ 4 & 3 & -6 \\ 5 & 4 & 0 \\ 6 & 5 & 0 \\ 7 & 4 & 0 \\ 8 & 5 & 0 \end{pmatrix}$$

2. 求解系统的状态方程和传递函数

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 “diag2ss.m”：

```
%清除所有内存变量
clear all;
%方框图初始化
n1=1;d1=1;
```

```
n2=0.5;d2=1;
n3=1;d3=[1 1];
n4=1;d4=[1 4];
n5=1;d5=[1 3];
n6=1;d6=[1 2];
n7=2;d7=2;
n8=1;d8=1;
%方框个数
nblocks=8;
%建立方框图
blkbuild;
%建立方框图的对角非连接形式的状态空间模型
sys=ss(a,b,c,d);
%建立方框图连接矩阵
q=[1 0 0;2 1 -8;3 2 -7;4 3 -6;5 4 0;6 5 0;7 4 0;8 5 0];
%求解方框图的状态空间模型
sysc=connect(sys,q,1,8);
%数据扩展
[A,B,C,D]=ssdata(sysc);
%将状态空间模型转换为传递函数模型
[num,den]=ss2tf(A,B,C,D,1);
%显示结果
disp('State-Space Model of the block-diagram is:');
A
B
C
D
disp('Transfer Function of the block-diagram is:');
num
den
```

选择"File"菜单的"Save"选项，保存文件名为"diag2ss.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 diag2ss，在 MATLAB Command Window 下查看运行的结果。

小结：在 connect ( ) 函数的基本调用格式 SYSc = CONNECT(SYS,Q,INPUTS,OUTPUTS) 中，SYS 是系统各框图无连接时的状态空间描述，Q 是系统的关联矩阵，INPUTS 和 OUTPUTS 分别是系统输入量和输出量所在方框的编号；SYSc 是返回的系统的状态方程描述。一般说来，在调用 connect ( ) 函数之前，首先要调用 blkbuid 语句使用，其功能是建立系统的方框图描述。blkbuid 语句是脚本（script）函数，其输入参数和返回值都不在函数体内说明。其输入参数有 nblocks，代表系统方框图的块数；（ni，di）或（ai，bi，ci，di），代表单个方框的传递

函数描述或状态方程描述。其返回值为（a，b，c，d），代表将这些方框对角连接后的系统状态方程描述（注意，并不是按原连接方式求出的系统状态方程描述，因为此时并未调用系统的关联矩阵）。

调用 blkbuild 语句时需要注意的是，因为输入参数在函数体外声明，所以应该在程序的最开始清除所有内存变量。否则即使在程序中改变了输入参数，也可能得到与未改变前相同的结果。事实上，调用 blkbuild 语句与使用 append（）函数的结果是一样的。append（）函数将系统的方框图转化为如下对角连接的形式：

$$A=\begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_n \end{pmatrix} \qquad B=\begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_n \end{pmatrix}$$

$$C=\begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_n \end{pmatrix} \qquad D=\begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{pmatrix}$$

然后再调用 connect（）函数，根据关联矩阵 Q 的信息求出系统的状态方程描述。程序中还用到了 ssdata（）函数，其功能是将系统的抽象描述 SYS 扩展成状态方程描述的系数矩阵。事实上，直接使用系统的抽象描述也可以得到希望的结果，例如执行完 diag2ss.m 文件后，在 MATLAB Command Window 下键入“sysc”，有

```
sysc
a =               x1        x2        x3        x4
         x1       -1        -1        -0.5      0
         x2       1         -4        0         -1
         x3       0         1         -3        0
         x4       0         0         1         -2
b =               u1
         x1       0.5
         x2       0
         x3       0
         x4       0
c =               x1        x2        x3        x4
         y1       0         0         1         0
d =               u1
         y1       0
Continuous-time system.
```

此（a，b，c，d）与结果中 ssdata（）扩展出的（A，B，C，D）是完全一致的。但需要注意的是，并不能直接调用系统的参数矩阵（a，b，c，d），这 4 个矩阵也不会覆盖程序中的同名变量。例如在 MATLAB Command Window 下键入“a”，有

```
a
a =  -1    0    0    0
      0   -4    0    0
      0    0   -3    0
      0    0    0   -2
```

这是 blkbuild 语句的返回值（a，b，c，d）中的 a 矩阵。如果希望使用系统的参数矩阵，还是要先用 ssdata（）语句进行数据扩展。

8.5 控制系统的稳定性

前面已经讲过如何在数学上描述控制系统并解出它的运动。只要知道了系统的结构和各参数，我们就能算出它的各物理量的变化规律。

从工程角度看，这还是不够的。一方面，系统越复杂，微分方程阶次就越高，求解也就越困难。对于实际工程领域的许多复杂的系统，微分方程阶次高达十几阶甚至几十阶，不用计算机实际上是无法求解的。另一方面，实际工程问题并不是简单地求解一个既定系统的运动，而往往是要选择系统中的某些参数，甚至还要改变系统的结构，以求获得较好的静态和动态性能。

对于上述这些问题，如果都靠直接求解微分方程来研究，势必要解大量的微分方程，从而大大增加计算量。同时，只从微分方程也不容易区分影响系统运动规律的主要因素和次要因素。

因此，就需要研究一些比较方便的工程分析方法。这些工程分析方法的计算量应当不太大，并且不因方程阶次的升高而增加太多。用这些方法不仅比较容易分析各主要参数对系统运动规律的影响，而且还可以借助一些图表和曲线直观地把运动特征表示出来。这些都是直接求解微分方程所做不到的。

在 MATLAB 环境下，虽然求解微分方程是非常容易的，但掌握有关系统稳定性的概念对于理论分析和工程实际应用来说都是非常必要而且是十分重要的。

关于运动稳定性严格定义有不只一种，但其中最重要的是还是俄国学者 A.M.李亚普诺夫于 1892 年提出的经典定义：

如果一个关于 x 的微分方程组，在初始条件 $x(t_0)=x_0$ 下有解 $x(t)$，且对于任意给定的正数 $\varepsilon>0$，总存在一个正数 $\delta(\varepsilon)$，当初始条件 x_0 变为 $\tilde{x}_0$ 时，只要 $\|\tilde{x}(t)-x(t)\|\leqslant\delta$，其相应解 $\tilde{x}(t)$ 在 $t>t_0$ 的任何时刻都满足 $\|\tilde{x}(t)-x(t)\|<\varepsilon$，则称解 $x(t)$ 是稳定的。如果不存在这样的正数 $\delta(\varepsilon)$，则称解 $x(t)$ 是不稳定的。

对于线性控制系统来说，其稳定性的充分必要条件是：它的微分方程的特征方程的全部根都是负实数或实部为负的复数，亦即：全部根都位于复数平面的左半平面。

如果特征方程在复数平面的右半平面上没有根，但在虚轴上有根，则可以说该线性系统是临界稳定的。

例如对于二阶系统： $$y=1-\frac{1}{\sqrt{1-\zeta^2}}\mathrm{e}^{-\zeta\omega t}\sin(\omega_d\omega t+\theta)$$

其中：　$\theta=\mathrm{arctg}\dfrac{\sqrt{1-\zeta^2}}{\zeta}$，$\omega_d=\sqrt{1-\zeta^2}$

当 0<ζ 时系统是稳定的，0<ζ<1 时系统的响应曲线振荡衰减；ζ>1 时系统响应曲线阻尼衰减；当 ζ=0 时系统临界稳定，系统响应曲线等幅振荡；ζ<-1 系统不稳定，系统的响应曲线是发散的，如图 8-20 所示。

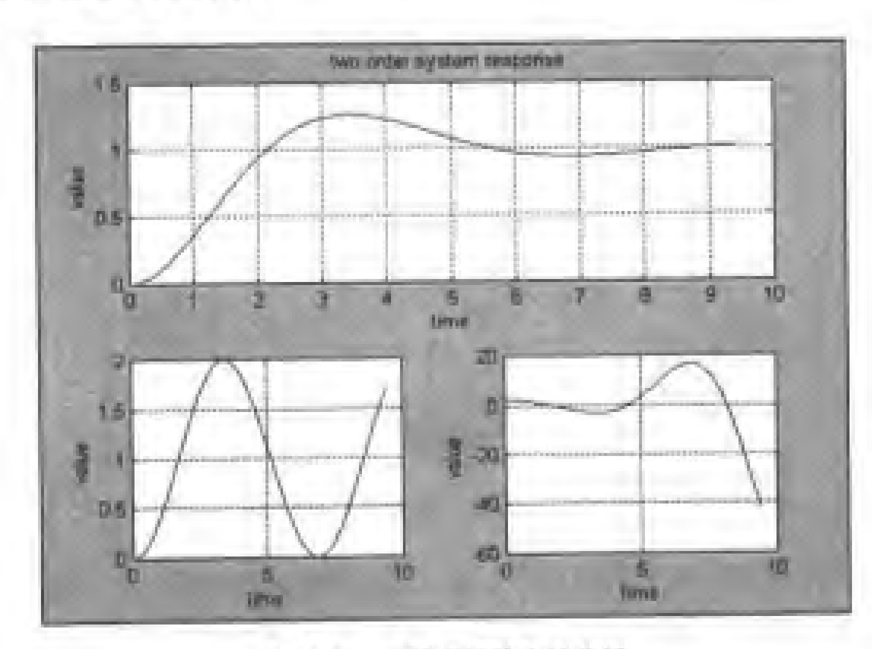


图 8-20　二阶系统稳定性分析

在计算机辅助设计软件出现之前，通常采用间接的方法判断系统的稳定性。例如 Routh 判据等等。在 MATLAB 环境下，我们不必再采用这样的方法，可以直接求解系统特征方程的根，根据根的分布来判断系统的稳定性。前边介绍过的 tf2zp（）函数、ss2zp（）函数等等都可以完成这项工作。如果仅仅知道系统的特征方程，还可以调用 roots（）函数来求解特征方程的根，然后再判断系统的稳定性。知道了系统的零点之后，还可以判断该系统是否为最小相位系统。所谓最小相位系统，就是系统的零点均在复平面的左半平面，当然，系统首先是稳定的。

某控制系统的状态方程描述如下。试判断其稳定性和是否为最小相位系统，并绘制其时间响应曲线来验证上述判断。

$$A=\begin{pmatrix}-3 & -8 & -2 & -4\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\qquad B=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$C=\begin{pmatrix}0 & 0 & 1 & 1\end{pmatrix}$$

结果：系统稳定并且是最小相位的。

z =　-1

p = -1.4737 + 2.2638i

```
    -1.4737 - 2.2638i
    -0.0263 + 0.7399i
    -0.0263 - 0.7399i
k = 1
```

系统的时间响应曲线如图 8-21 所示。

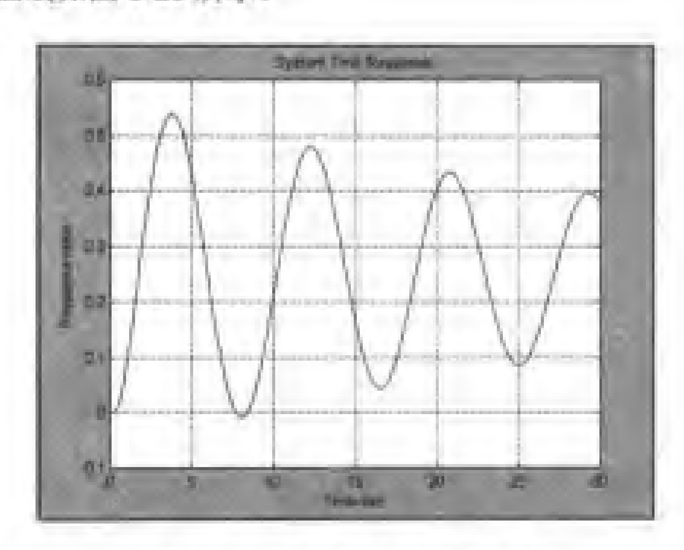


图 8-21　线性系统时间响应曲线

分析：系统的极点均在复平面的左半平面，因而系统是稳定的。从系统的时间响应曲线上看也是如此。系统的输出振荡衰减。但可以看出，系统衰减响应曲线的振荡频率很高，衰减速率很慢。这是因为系统有两个极点的实部为-0.0263，非常靠近虚轴的缘故。在过程上，这种系统属于性能非常差的一类。稳定裕量很小，稍有干扰系统的极点就可能越过虚轴跑到复平面的右半平面去。从而造成系统的不稳定。对于这类系统，一般都要加校正手段，使其极点远离虚轴。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M 文件，进入 MATLAB Editor/Debugger，编辑 M 文件“teststable.m”：

```
%系统状态方程模型
A=[-3 -8 -2 -4;1 0 0 0;0 1 0 0;0 0 1 0];
B=[1;0;0;0];
C=[0 0 1 1];
D=[0];
%标志变量，判断是否稳定
flag1=0;
%标志变量，判断是否最小相位
flag2=0;
%求解零极点和增益
[z,p,k]=ss2zp(A,B,C,D,1);
%显示结果
```




```
disp('System zero-points,pole-points and gain are:');
z
p
k
%判断是否稳定
n=length(A);
for i=1:n
    if real(p(i))>0
        flag1=1;
    end
end
%判断是否最小相位
m=length(z);
for i=1:m
    if real(z(i))>0
        flag2=1;
    end
end
%显示结果
if flag1==1
    disp('System is unstable');
else
    disp('System is stable');
end
if flag2==1
    disp('System is Nonminimal Phase');
else
    disp('System is minimal Phase');
end
%系统仿真变量初始化
x0=[0 0 0 0];
t=0:0.1:30;
r=length(t);
u=ones(1,r);
[y,T]=lsim(A,B,C,D,u,t,x0);
%图形绘制
plot(t,y);
title('System Time Response');
xlabel('Time-sec');
```

```
ylabel('Response-value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“teststable.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 teststable，在 MATLAB Command Window 下查看运行的结果。

小结：本例使用了 lsim ( ) 函数进行线性系统仿真，也可以用求解系统状态方程的方法进行系统仿真，效果是一样的。本例还使用了 ones ( ) 函数，其功能是产生任意维数的元素值均为 1 的矩阵。

判断非线性系统的稳定性时经常要用到李亚普诺夫稳定性基本定理，但该定理的陈述不是构造性的，需要使用者自己去寻找满足条件的李亚普诺夫函数。这一过程是相当困难的，在这方面目前的控制系统计算机辅助设计软件也没有提供什么得力的工具。不过在线性定常系统方面，MATLAB 提供了一个求解李亚普诺夫方程（也称塞尔维斯特（Sylvester）方程）的函数 lyap（），可以用来求解形如：

$$A^T P + PA = -Q$$

的方程。其中 Q 是任意正定矩阵，如果能找到正定矩阵 P 满足上述方程，则该线性定常系统的平衡态是渐进稳定的。例如系统状态方程矩阵为

$$A=\begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}$$

则在 MATLAB Command Window 下键入

```
A=[0 1;-1 -1];
Q=[1 0;0 1];
P=lyap(A,Q)
```

求得的正定矩阵 P 为：

```
P = 1.5000    -0.5000
   -0.5000    1.0000
```

P 有解，说明该系统的平衡态是渐进稳定的。这里为了方便选 Q 矩阵为单位阵，当然也可以选别的正定矩阵。

有关控制系统的数学描述就讨论到这里，本章我们主要讨论了 MATLAB 环境下控制系统运动方程、控制系统的传递函数描述、控制系统的状态方程描述以及各个控制系统模型之间的转换问题，并在此基础上，给出了控制系统稳定性的判据和数值检验方法。

其中控制系统运动方程即微分方程描述是控制系统数学描述的基础，其他各种描述形式都是在此基础上推演或发展起来的。虽然目前在工业领域使用较多的是控制系统的传递函数描述和状态方程描述这两种手段，但对于一个完全陌生的控制系统来说，还是要首先分析其机理，列出运动方程，然后才能化为传递函数描述或状态方程描述。从下一章开始将介绍有关 MATLAB 环境下控制系统的时频响应分析。

第9章 控制系统时频分析及根轨迹的绘制

描述动态系统运动有两类方法。其中一类是用微分方程描述，状态方程也包括在内；另一类是用传递函数描述，传递函数矩阵是它的一种推广，框图是它的一种图解。

有些情况下也常常用典型响应描述一个动态系统的性质。所谓响应，就是指由于输入量的作用而造成的对象输出量的变化的函数。典型响应是指零初值条件下某种典型的输入量函数作用下对象的响应。

当我们借助于传递函数的概念建立了：

$$y(s)=G(s)u(s)$$

这样的关系式以后，就拉普拉斯变换的像函数而言，输出量 $y(s)$ 与输入量 $u(s)$ 之间好像有了一种“比例”关系，其“比例系数”就是传递函数 $G(s)$。这样就很容易想到：能否以某种“单位”输入量信号下的输出量函数表示传递函数，从而以一种直观的方式表达系统的特性。这就是典型响应的概念。

常用的典型形象有脉冲响应、阶跃响应和频率响应。在这一章的时频分析中我们将详细讨论这三种响应。

我们知道，一个控制系统的全部性质，都取决于其闭环传递函数：稳定性取决于其极点；静态精度取决于其增益；动态性能即取决于其极点，也与其零点有关。我们又知闭环传递函数的零点与开环传递函数的零点相同，增益之间也有简单的关系，都不难确定。惟有闭环传递函数的极点，即闭环特征方程的根，手工计算起来比较困难。根轨迹法是 W. R. Evans 于1948年提出的一种求解闭环特征方程根的简便的图解方法，在工程上获得了广泛的应用。他根据系统开环传递函数极点和零点的分布，依照一些简单的规则，用作图的方法求出闭环极点的分布，从而避免了复杂的数学计算。系统中某个参数的变化对闭环极点的分布会产生什么影响，可以很容易地从图上看出来。虽然在 MATLAB 环境下可以很方便的求出闭环传递函数特征方程的根，但掌握根轨迹的规则和绘制方法对于控制系统的设计和校正还是十分必要的。

9.1 时域响应分析

在上一章里，我们曾经介绍了 MATLAB 环境下微分方程的描述和解法。事实上，所谓控制系统的时域响应分析就是在时间域内求解系统的微分方程，然后根据绘制出来的曲线分析

系统的性能和各主要参数对系统性能的影响。不过这里的响应曲线一般是指典型的响应曲线，即所谓阶跃响应和脉冲响应。其激励函数分别为单位阶跃函数 $u(t)$（或 $l(t)$）和单位脉冲函数 $\delta(t)$。单位阶跃函数的数学表达式为：

$$u(t)=\begin{cases}0 & (t<0)\\ 1 & (t\geqslant 0)\end{cases}$$

对应拉普拉斯变换为 l/s。单位脉冲响应的数学表达式为：

$$\delta(t)=\begin{cases}0 & (t\neq 0)\\ \infty & (t=0)\\ \int_{-\infty}^{\infty}\delta(t)\,\mathrm{d}t=1\end{cases}$$

对应的拉普拉斯变换为 1。

求解系统典型响应的思路是首先列出微分方程，对方程两边同时取拉普拉斯变换得到系统的传递函数，然后根据输入量的类型确定输出量的表达式，求解此表达式并编制绘图程序，最后根据绘制出来的图形对系统进行分析。事实上，MATLAB 提供了一些函数可以直接求解系统的典型响应，可以让使用者从上述复杂的程序编制过程中解放出来，把更多的精力放在系统性能分析上。

例如,MATLAB 控制系统工具箱提供了一条求解系统单位阶跃响应的函数 step（），其基本调用格式有：

STEP(SYS1,SYS2,...,T) 或 [Y,T,X] = STEP(SYS,...)

两种。其中第一种格式是不关心系统阶跃响应的具体数值，仅仅在一张图上绘制从系统 SYS1 到系统 SYSn 的阶跃响应曲线，T 是 t0：tspan：tfinal 格式的时间向量。第二种格式返回系统的阶跃响应数据，但并不在屏幕上绘制系统的阶跃响应曲线。其中 T 是返回的仿真时间向量，设其维数为 LT，并假设系统有 Nu 维输入量，Ny 维输出量，则返回值 Y 是 LT×Nu×Ny 维矩阵，其 Y（：,：，j）行向量表示第 j 个输入通道的时间响应数据；对于含有 Nx 个状态变量的系统的状态方程描述，X 是 LT×Nx×Ny 维矩阵，对应各状态变量的阶跃响应数据。在这两种格式中，SYS 可以是系统的传递函数描述[NUM，DEN]，NUM 和 DEN 分别代表传递函数的分子和分母多项式系数的降幂排列；也可以是系统的状态方程描述（A，B，C，D）；还可以是系统的抽象描述。

对于系统的单位脉冲响应，MATLAB 也提供了一条函数 impulse（）实现此功能。其基本调用格式也是 IMPULSE(SYS1,SYS2,...,T) 或 [Y,T,X] = IMPULSE(SYS,...)两种，具体参数和返回值的含义与 step（）函数相同。请看下例：

系统的传递函数如下。试求其闭环传递函数，并绘制输出量阶跃响应曲线和脉冲响应曲线。选择函数的状态变量将其化为状态方程模型，并绘制状态变量的阶跃响应曲线和脉冲响应曲线

$$G_o(s)=\frac{200}{s^4+20s^3+140s^2+400s+384}$$

结果：系统的闭环传递函数系数为

```
numc = 0      0      0      0    200
denc =  1     20    140    400   584
```

$$G_c(s)=\frac{200}{s^4+20s^3+140s^2+400s+384}$$

系统的状态方程模型为

```
A =   -20   -140   -400   -584
        1      0      0      0
        0      1      0      0
        0      0      1      0
B =     1
        0
        0
        0
C =     0      0      0    200
D =     0
```

系统输出量的阶跃响应如图 9-1 所示。其横坐标为时间，纵坐标为系统的输出量阶跃响应值。

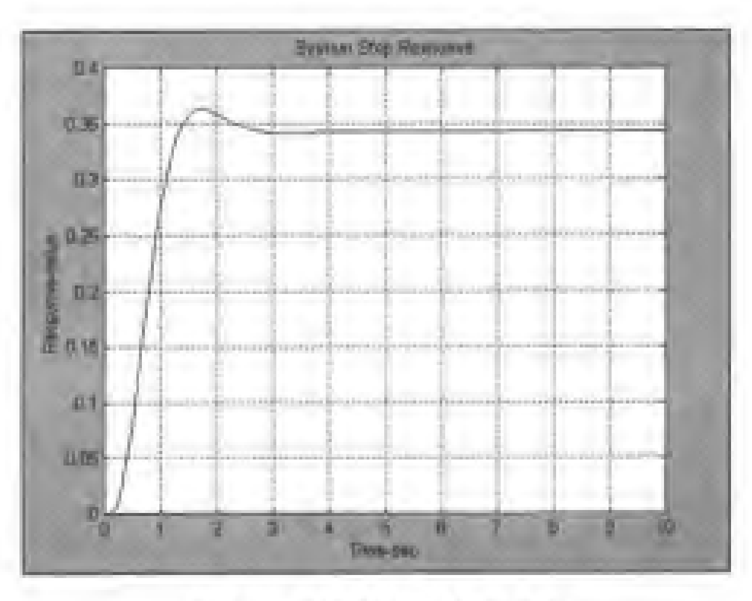


图 9-1　系统输出量的阶跃响应

系统输出量的脉冲响应如图 9-2 所示。其横坐标为时间，纵坐标为系统的输出量脉冲响应值。

系统状态变量的脉冲响应曲线如图 9-3 所示。其横坐标为时间，纵坐标为系统的状态变量阶跃响应值。因为本系统有 4 个状态变量，所以用四种不同描点格式来显示它们的响应值。

系统状态变量的脉冲响应曲线如图 9-4 所示。其横坐标为时间，纵坐标为系统的状态变量脉冲响应值。同样用四种不同描点格式来显示它们的响应值。

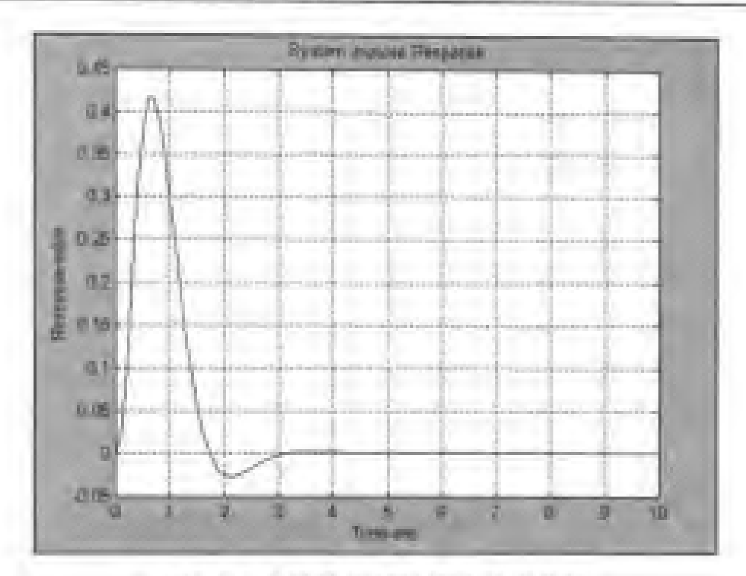


图 9-2　系统输出量的脉冲响应曲线

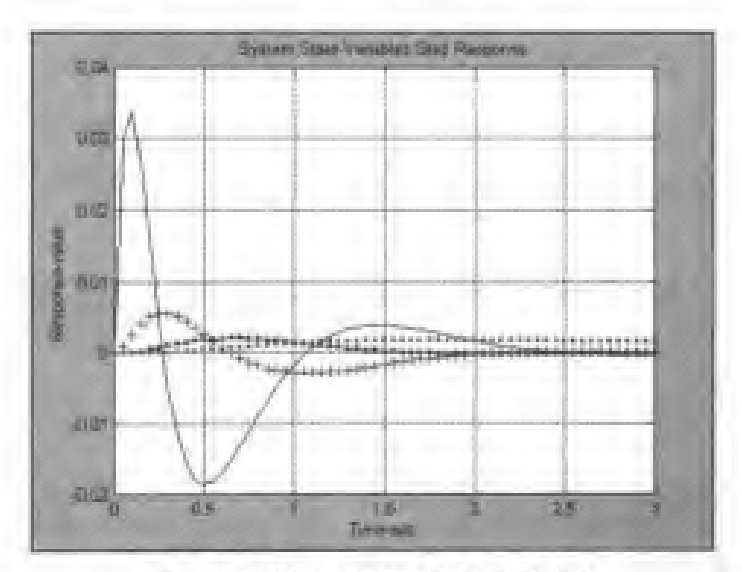


图 9-3　系统状态变量的阶跃响应曲线

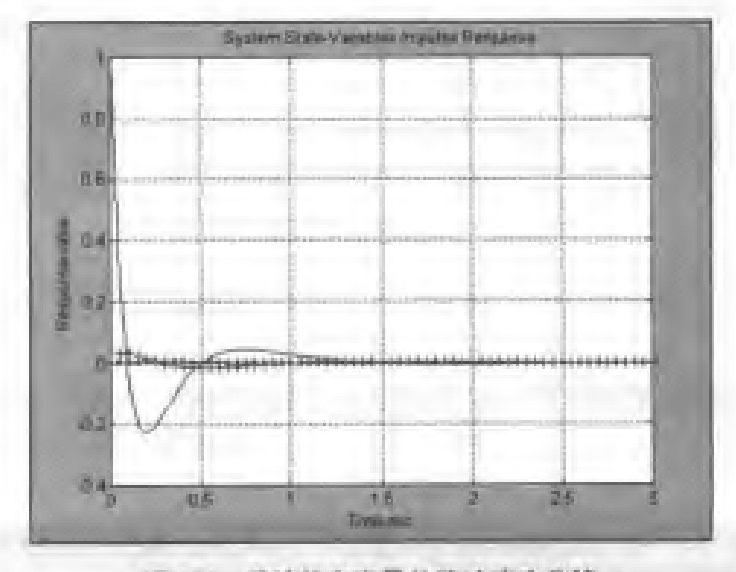


图 9-4　系统状态变量的脉冲响应曲线

分析：在单位负反馈情况下，系统的闭环传递函数表达式为 G(s)/(1+G(s))。因此，系统闭环传递函数的分子多项式与开环传递函数的分子多项式相同；闭环传递函数的分母多项式是开环传递函数的分母多项式加分子多项式。

从系统输出量的阶跃响应来看，响应曲线振荡幅度很小，也就是超调量比较小，大约 0.02/0.34=6%左右。并且达到峰值后迅速下调，在 3s 附近已经基本达到稳态值，也就是过渡过程时间不超过 3s。

从系统输出量的脉冲响应曲线上来看也可以验证这些说法：脉冲响应曲线大约在 1.7s 左右过零，此时对应阶跃响应曲线的峰值点；脉冲响应曲线只有一个过零点，对应阶跃响应曲线只振荡一次就达到稳态值；并且脉冲响应曲线也是在大约 3s 左右达到稳态值 0。

因为系统闭环传递函数的特征方程是 4 阶的，所以化成状态方程模型后有四个状态变量。对应系统状态变量的阶跃响应和脉冲响应就各有 4 条曲线。其中实线对应第一个状态变量，"＋"对应第二个状态变量，"－"对应第三个状态变量，"."对应第四个状态变量。这些状态变量的阶跃响应曲线和脉冲响应曲线都是经过短暂振荡后迅速回到零点，这从另一个侧面验证了上面的那些说法。其中第一个状态变量的振荡的峰值最大，次数也最多。这是因为 tf2ss（）函数将系统的传递函数模型转化为状态空间的可控规范型，第一个状态变量与系统输入量 u（t）最接近。零时刻输入量加入到系统中，引起状态变量的突然变化。对于其他的状态变量来说，它们和输入量 u（t）之间隔着 1 阶～3 阶的惯性环节，阶跃函数和脉冲函数的冲激效应就缓和许多了。因此振荡的峰值就比较小。

一般说来，具有上述这些性质的系统是我们在控制系统中希望看到的，即系统输出的超调量很小，过渡过程时间很短，能够迅速跟踪输入量设定值的变化。当然，这一切都是在系统稳定的前提下而言的。另外，本例系统的阶次比较高，容易引进高频干扰。实际应用中一般用二阶模型近似后再进行处理。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"sysresponse.m"：

```
%关闭所有图形窗口并清除内存变量
close all;
clear;
%系统开环传递函数初始化
numo=[0 0 0 0 200];
deno=[1 20 140 400 384];
%求解系统的闭环传递函数
numc=numo;
n=length(deno);
denc=zeros(1,n);
denc=numo+deno;
%结果显示
disp('System Closed Loop Transfer Function is:')
numc
```

```
denc
%系统仿真数据初始化
t=0:0.05:3;
%系统输出量的阶跃响应
y=step(numc,denc,t);
%系统输出量的脉冲响应
yy=impulse(numc,denc,t);
%输出量阶跃响应曲线绘制
plot(t,y);
title('System Step Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
%输出量脉冲响应曲线绘制
plot(t,yy);
title('System Impulse Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
%求解系统的状态方程模型
[A,B,C,D]=tf2ss(numc,denc);
%状态方程模型结果显示
disp('System State-Space Model is:');
A
B
C
D
%系统状态变量的阶跃响应
[ys,x]=step(A,B,C,D,1,t);
%系统状态变量的脉冲响应
[yys,xx]=impulse(A,B,C,D,1,t);
%状态变量节约响应曲线绘制
plot(t,x(:,1),t,x(:,2),'+',t,x(:,3),'.-',t,x(:,4),'.')
title('System State-Variables Step Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
%状态变量脉冲响应曲线绘制
plot(t,xx(:,1),t,xx(:,2),'+',t,xx(:,3),'.-',t,xx(:,4),'.')
```

```
title('System State-Variables Impulse Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“sysresponse.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysresponse，在 MATLAB Command Window 下查看运行的结果。

小结：本例分别调用了两种不同格式的 step（）函数和 impulse（）函数，并且根据不同的系统描述模型绘制了输出量和状态变量的阶跃响应和脉冲响应。一般说来，在控制系统仿真和设计的时间域分析有这两种响应曲线就足够了，系统的主要参数及其变化规律在这两种响应曲线中都有所反映。

从工程的角度来看，实际控制系统的输入函数千变万化，实际模拟控制系统时仅靠上述这两种曲线虽然也是一种解决问题的思路（从理论上说，阶跃函数和脉冲函数之和可以以任意精度逼近某一函数），不过计算的复杂性就大大增加了。因此，MATLAB 还提供了一个函数可以求解和绘制任意输入函数激励的系统时间响应，这就是前文介绍过的 lsim（）函数。例如希望求解传递函数为

$$G(s)=\frac{s+1}{s^3+7s^2+14s+8}$$

的控制系统在输入函数 u=sint 时的时间响应，可以在 MATLAB Command Window 下键入下列语句：

```
num=[0 0 1 1];
den=[1 7 14 8];
t=0:0.05:5;
u=sin(t);
lsim(num,den,u,t)
```

系统的响应曲线如图 9-5 所示。

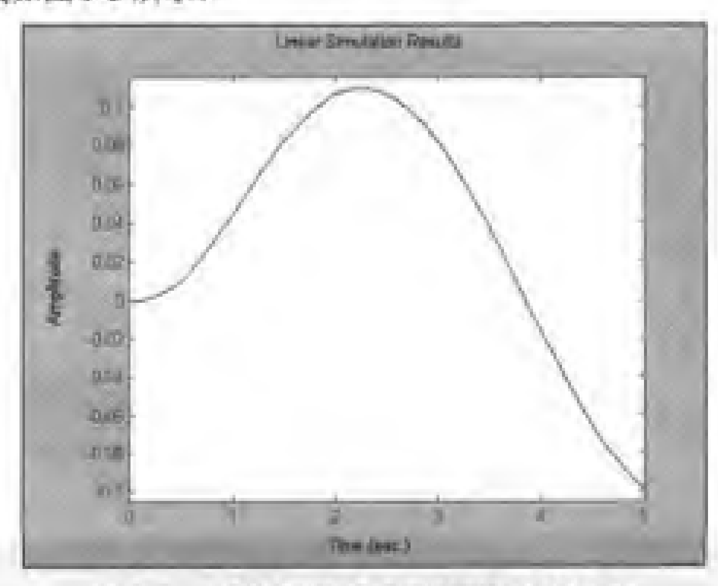


图 9-5　正弦函数激励下三阶线性系统的响应

其横坐标为时间，纵坐标为系统在单位正弦函数 u=sint 激励下的响应值。从图中可以看出，大约在 2.3s 左右系统响应曲线达到最大值，这也是系统在单位正弦激励下的共振点。本书第 4 章对系统进行频率分析时将具体讲述这方面内容。

当然，在 MATLAB 环境下也可以调用[Y,T] = LSIM(SYS,U,...)格式的 lsim（）函数，根据返回的时间向量 T 和任意函数响应值 Y，通过 plot（）语句绘制任意函数激励的响应曲线。这样的好处是可以随意控制曲线的绘制格式。

对于离散时间的控制系统（或称采样控制系统），MATLAB 同样也提供了相应的 dstep（）、dimpulse（）和 dlsim（）函数求解其单位阶跃响应、单位脉冲响应和任意函数的激励响应。

这三条函数的参数设置、返回值和调用格式与连续控制系统相应的 step（）、impulse（）和 lsim（）函数完全一致，只不过将相应的传递函数描述和状态方程描述换为脉冲传递函数描述和离散状态方程描述，并且求得的响应数据也是离散的。例如希望求解离散控制系统：

$$\overline{x}(z)=\frac{1.6z^2-z}{z^2-0.8z+0.5}$$

的单位阶跃响应和单位脉冲响应，可以在 MATLAB Command Window 下键入下列语句：

```
num=[1.6 -1 0];
den=[1 -0.8 0.5];
subplot(211),
dstep(num,den);
subplot(212),
dimpulse(num,den);
```

此离散系统的响应曲线如图 9-6 所示。

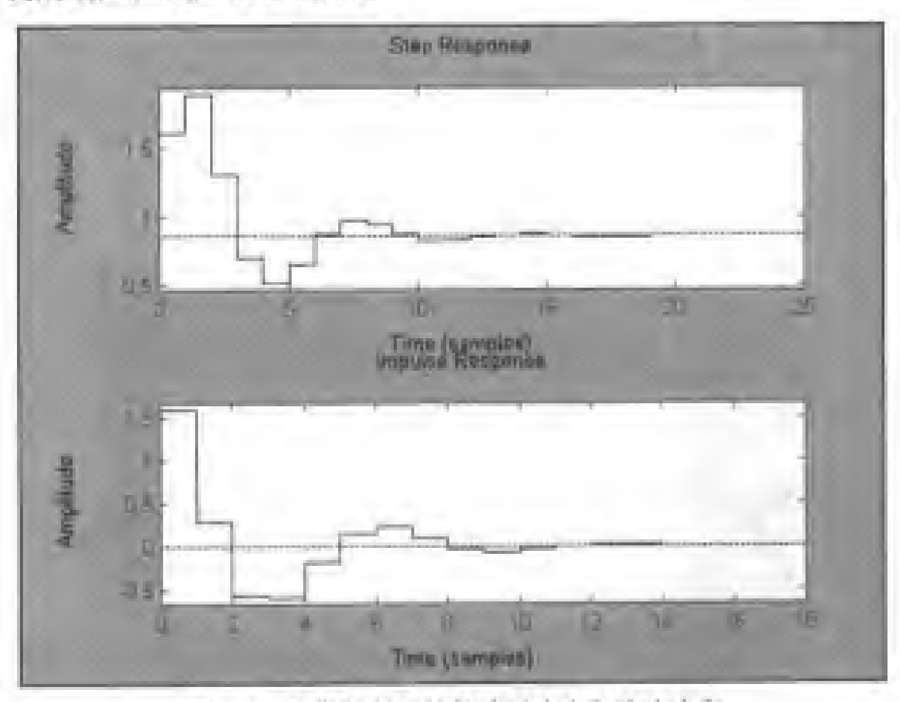


图 9-6　离散控制系统的阶跃响应和脉冲响应

其横坐标均为时间采样值（samples），上半部分的纵坐标为离散系统的阶跃响应值，下半部分为离散系统的脉冲响应值。从曲线的形状来看，趋势与连续系统响应基本类似，都是趋于

设定值。但由于是离散系统，每个采样时间内都有一个一阶保持器，因此曲线的外型呈台阶状。

9.2　频率响应分析

我们已经知道，直接用微分方程研究控制系统虽然可以准确地解出系统的运动函数，然而从工程角度看，这种方法既嫌不足，又嫌多余。人们所期望的工程研究方法是：计算量不应当太大，而且计算量不应随微分方程的阶次升高而增加太多；容易区分出系统运动的主要因素以及各因素对系统总体动态性能的影响。以上这些要求，是直接用微分方程研究系统的方法所难以做到的。本节将要讲述的频率响应法和下一节将要讲述的根轨迹法正是满足这些要求的很好的工程研究方法。

频率响应法的基本思想是把控制系统中的各个变量看成一些信号，而这些信号又是由许多不同频率的正弦信号合成的；各个变量的运动就是系统对各个不同频率的信号的响应的总和。

这种观察问题和处理问题的方法起源于通信科学。20 世纪 30 年代，这种观点被引进控制科学，对控制理论的发展起了强大的推动作用。它克服了直接用微分方程研究控制系统的种种困难，解决了许多理论问题和工程问题，迅速形成了分析和综合控制系统的一整套方法，即频率响应法。英国的剑桥学派又将频率响应法推广到多变量系统，因此这种方法直到今日仍然是控制理论中极为重要的基本内容。

频率响应法之所以能发挥这样的作用，是因为它具有一系列重要的优点。首先，这种方法物理意义鲜明。按照频率响应的观点，一个控制系统的运动无非是信号在一个一个环节之间依次传递的过程；每个信号又是一些不同频率的正弦信号合成的；这些不同频率的正弦信号的振幅和相角在传递过程中，依一定的函数关系变化，就产生形式多样的运动。这种观点比简单的把控制系统观念看成一个微分方程显然更容易理解，并且更能启发人们区分影响系统的主要因素和次要因素，进而考虑改善系统性能。其次，从信号传递的角度出发，可以用实验方法求出对象的数学模型，这一点在工程上价值很大，特别是对于机理复杂或机理不明而难以列写微分方程的对象，频率响应观点揭示了重要的处理方法。第三，对于手工计算来说，频率响应法的计算量小。用它分析系统的运动与直接求解系统的微分方程式相比，所需的手工计算量相差非常悬殊。第四，由于频率响应法很大一部分都采用作图，因此这种方法有很强的直观性。

当然，频率响应法不能用于对非线性系统进行全面分析，尽管它在这方面也获得了一定的成绩。因为非线性系统不满足叠加原理，所以从根本上说频率响应法不可能成为研究和设计非线性系统的得力工具。这是它主要的局限性。

对于 MATLAB 环境下控制系统的仿真和设计来说，虽然频率域分析和时间域分析在编程的工作量方面没有太大的区别，但采用频率响应的观点无疑能够更加清晰地了解系统的本质。尤其是对于系统的传递函数描述，采用频率响应法分析具有得天独厚的优势，只需将传递函数的自变量 s 换成 jω 即可。

9.2.1　频率响应

从形式上来看，频率响应法用到的频率特性函数与系统的传递函数相同，只是把自变量

s 换成 $\mathrm{j}\omega$。对于输入量为 y，输出量为 u 的系统来说，其频率特性函数为

$$G(\mathrm{j}\omega)=\frac{\dot{Y}}{\dot{U}}=G(s)\Big|_{s=\mathrm{j}\omega}$$

不过从概念上看，系统的传递函数和频率特性函数还是有一定区别的。严格地说，系统传递函数的自变量 $s=\sigma+\mathrm{j}\omega$，其中 σ 和 ω 都是实数。事实上，频率特性函数就相当于传递函数的自变量 s 只沿复数屏幕的虚轴变化。正因为如此，传递函数和频率特性函数习惯上总是用同一字母 G 表示。

频率特性函数 $G(\mathrm{j}\omega)$ 是依赖于 ω 的函数。在给定的 ω 下，它就是一个复数。习惯上用 $|G(\mathrm{j}\omega)|$ 表示函数 $G(\mathrm{j}\omega)$ 的模，它也是 ω 的函数，称为幅频特性函数。$\arg G(\mathrm{j}\omega)$ 表示函数 $G(\mathrm{j}\omega)$ 的相角，它也是 ω 的函数，称为相频特性函数。幅频特性函数表示的是正弦输出信号与正弦输入信号的振幅之比，而相频特性函数表示的是正弦输出信号相对于正弦输入信号相位的领先。因此，有：

$$\begin{cases}|G(\mathrm{j}\omega)|=\dfrac{|\dot{Y}|}{|\dot{U}|}\\ \arg G(\mathrm{j}\omega)=\arg\dot{Y}-\arg\dot{U}\end{cases}$$

推导系统频率特性函数的数学基础是非周期函数的傅里叶变换（Fourier Transform），相关知识在低年级的工科数学课程中已经有详细介绍，这里就不再赘述了。

正因为系统的传递函数和频率特性函数在表达式上完全相同，因此在 MATLAB 环境下对此二者是不加以区分的。处理和求解系统传递函数的语句和函数对频率特性函数同样适用。但我们在使用过程中应该注意到，传递函数的观点和系统频率响应的观点是两种不同的认识控制系统的思路。在 MATLAB 环境下求解系统的幅频特性函数和相频特性函数时，经常要用到 polyval（）函数。其功能是求解某一给定系数和自变量的多项式的值。其基本调用格式为

$$Y=\mathrm{POLYVAL}(P,X)$$

其中 P 是多项式系数，X 是自变量，Y 是求得的返回值。请看下例：

某控制系统的频率特性函数如下。试求解其幅频特性曲线和相频特性曲线，并与系统的阶跃响应曲线加以对比。

$$G(\mathrm{j}\omega)=\frac{7(\mathrm{j}\omega)}{(\mathrm{j}\omega)^3+4(\mathrm{j}\omega)^2+8(\mathrm{j}\omega)+9}$$

结果：系统的幅频特性曲线和相频特性曲线如图 9-7 所示。

分析：从系统的频率特性响应曲线来看，幅频特性曲线的谐振峰比较小，低通特性很好；系统相频特性曲线的起始阶段也基本是直线，后边的跳变是因为反正切函数的特性。从系统的频率特性响应曲线上，我们还可以得出对应频率下系统的增益和相位值。对应系统的阶跃响应曲线，系统的输出量变化比较平缓，超调量比较小，振荡次数也不多，很快就达到了稳态值。因此，从系统的频率响应特性曲线我们基本上能够预测系统时间域阶跃响应的情况，并且还可以推断任意激励函数下的响应。

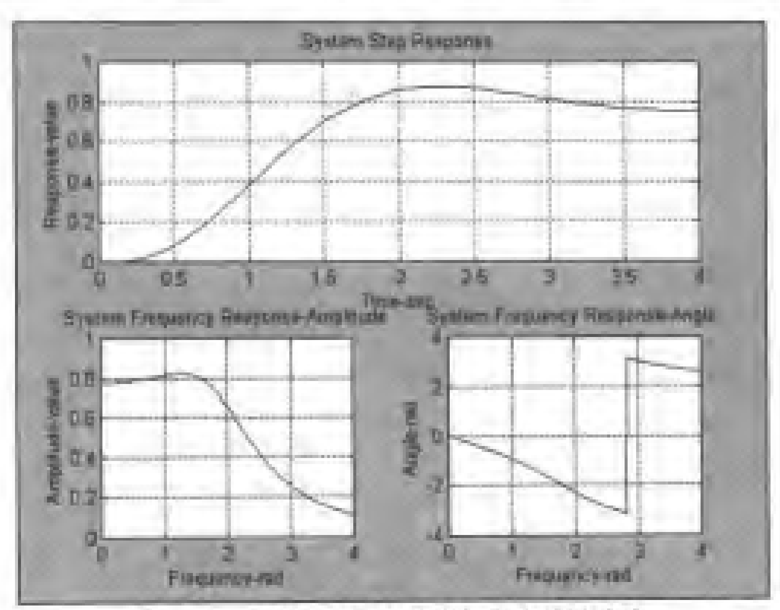


图 9-7　系统阶跃响应曲线与频率特性曲线

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"freqresponse.m"：

```
%关闭所有窗口并清除所有内存变量
clear all;
close all;
%系统的频率特性函数描述
num=[0 0 0 7];
den=[1 4 8 9];
%仿真时间和频率初始化
```

图 9-7 的上半部分是系统的阶跃响应曲线，横坐标是时间，纵坐标是阶跃响应的值；下半部分是系统的频率响应曲线，横坐标都是频率；其中左边是相频特性曲线，右边是幅频特性曲线。

```
t=0:0.05:4;
wt=0:0.02:4;
%系统的阶跃响应
y=step(num,den,t);
%系统的频率特性
G=polyval(num,sqrt(-1)*wt)./polyval(den,sqrt(-1)*wt);
%幅频特性
mag=abs(G);
%相频特性
theta=angle(G);
%绘图
subplot(211),
```

```
plot(t,y);
title('System Step Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
subplot(223),
plot(wt,mag);
title('System Frequency Response-Amplitude');
xlabel('Frequency-rad');
ylabel('Amplitude-value');
grid;
subplot(224),
plot(wt,theta);
title('System Frequency Response-Angle');
xlabel('Frequency-rad');
ylabel('Angle-rad');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“freqresponse.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 freqresponse，在 MATLAB Command Window 下查看运行的结果。

小结：polyval（）函数的自变量可以是复数 sqrt（-1），这就给求解系统的频率特性函数带来了很大的方便。另外，本例绘制的幅频特性曲线和相频特性曲线是直接采用的直角坐标，没有采用习惯的对数或半对数坐标的波特（Bode）图。关于系统频率特性波特图的绘制，我们将在下一节介绍。

当然，freqs（）函数同样可以求解频率特性函数的响应曲线，其调用格式与参数是传递函数时一致。请看下例：

某三阶系统的传递函数如下。试根据其主导极点给出其低阶近似的频率特性函数，并比较原系统与低阶系统的阶跃响应和频率响应。

$$G(s)=\frac{750}{s^3+36s^2+205s+750}$$

结果：系统的极点为

```
ans = -30.0000
      -3.0000 + 4.0000i
      -3.0000 - 4.0000i
```

相应频率特性函数为

$$G(\mathrm{j}\omega)=\frac{25}{(\frac{\mathrm{j}\omega}{30}+1)(\mathrm{j}\omega+3+4\,\mathrm{j})(\mathrm{j}\omega+3-4\,\mathrm{j})}$$

可以看出，低频段该函数分母的第一项将非常接近于 1。考虑到系统的低通特性，可以



认为极点-30 远离原点，对低频段影响很小。系统的主导极点为-3±4j，对应简化低阶模型为：

$$G(j\omega)=\frac{25}{(j\omega)^2+6(j\omega)+25}$$

原系统和低阶近似系统的阶跃响应如图 9-8 所示。

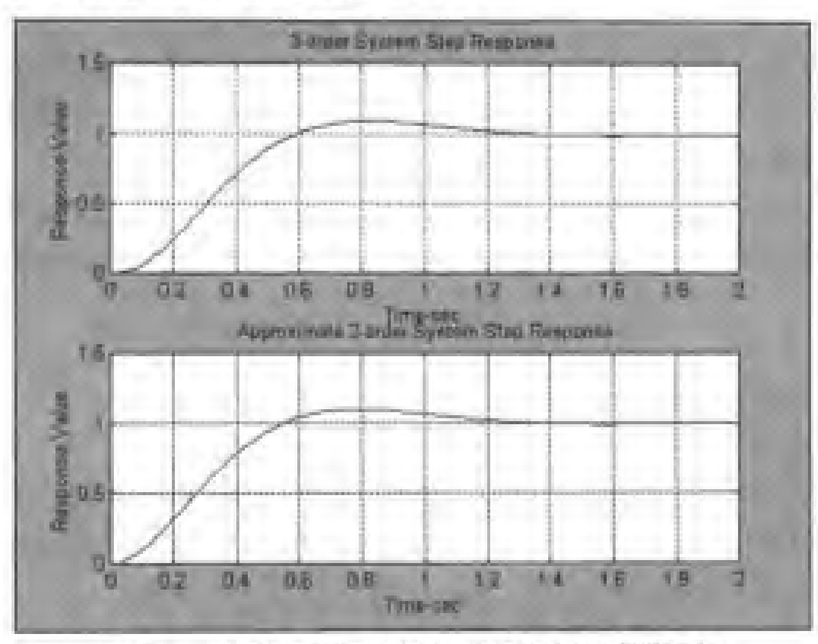


图 9-8　原系统及其低阶近似系统阶跃响应曲线比较

图 9-8 上半部分是原系统的阶跃响应曲线，下半部分是二阶近似系统的阶跃响应曲线。横坐标均为时间，纵坐标均为响应值。

原系统与低阶近似系统的频率响应如图 9-9 所示。

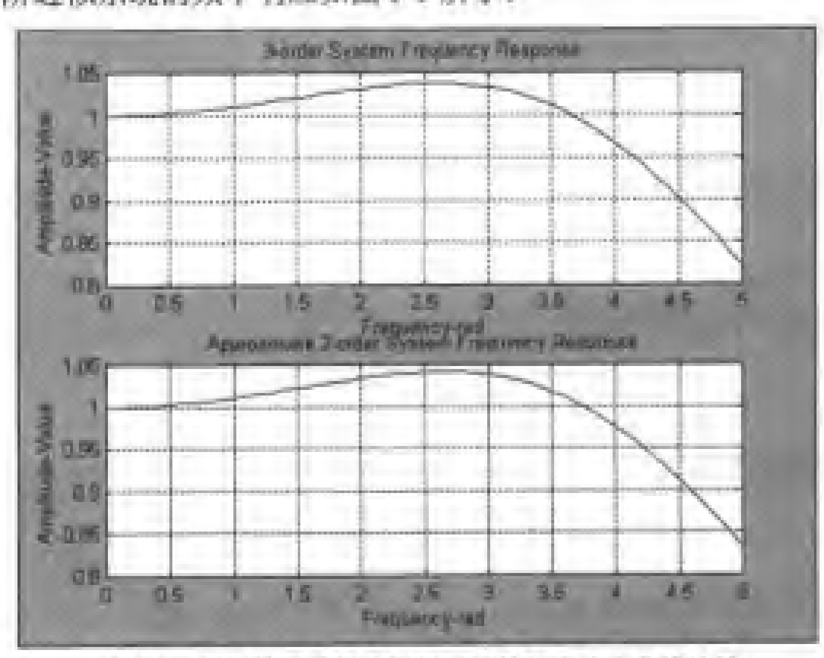


图 9-9　原系统及其低阶近似系统的频率响应曲线比较

其上半部分是原系统的频率响应曲线，下半部分是二阶近似系统的频率响应曲线。横坐标均为频率，纵坐标均为响应值。

分析：在实际控制系统分析过程中，寻找高阶系统的低阶近似是一种非常普遍的思路。通常采用的方法除了本例的求解系统主导极点外，还有近似零极相消等方法。所谓主导极点，是指稳定的高阶系统中离虚轴最近的一对共轭极点，附近没有零点，并且其他极点距这对极点很远或附近有零点与其相消。原高阶系统的性质与使用这对主导极点近似的二阶系统的性质大体相同。

从本例的结果图中我们也可以看出，原系统的阶跃响应和频率响应与其二阶近似系统几乎一模一样，只是在响应的起始阶段原系统的曲线更加弯曲。显然，这是因为多了一个惯性环节的原因。在控制系统机理复杂难于分析或阶次较高的情况下，也可以先用实验的方法求出系统的一些基本参数，例如谐振峰大小和位置、截止频率、超调量和过渡过程时间等等，然后用合适的低阶系统（一般是二阶系统，因为人们对二阶系统研究得最透彻）来近似原系统。

求解过程：

本例的解题步骤分为两部分：

(1) 求解系统的极点

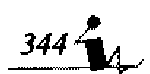
因为本例只需求解系统的极点，所以不必调用求解零点、极点和增益的 tf2zp（）函数，只需调用根据多项式系数求根的 roots（）函数即可。在 MATLAB Command Window 下键入下列语句：

```
den=[1 36 205 750];
roots(den)
```

(2) 求解原系统及其低阶近似系统的阶跃响应和频率响应

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 “sysapprox.m”：

```
%关闭所有窗口并清除内存变量
clear all;
close all;
%原系统的频率特性函数
num1=[0 0 0 750];
den1=[1 36 205 750];
%低阶近似系统的频率特性函数
num2=[0 0 25];
den2=[1 6 25];
%仿真时间及频率初始化
t=0:0.02:2;
wt=0:0.1:5;
%原系统的阶跃响应和频率响应
y1=step(num1,den1,t);
g1=freqs(num1,den1,wt);
amp1=abs(g1);
%低阶近似系统的阶跃响应和频率响应
```

```
y2=step(num2,den2,t);
g2=freqs(num2,den2,wt);
amp2=abs(g2);
%图形绘制
subplot(211),
plot(t,y1);
title('3-order System Step Response');
xlabel('Time-sec');
ylabel('Response-Value');
grid;
subplot(212),
plot(t,y2);
title(' Approximate 2-order System Step Response');
xlabel('Time-sec');
ylabel('Response-Value');
grid;
subplot(211),
plot(wt,amp1);
title('3-order System Frequency Response');
xlabel('Frequency-rad');
ylabel('Amplitude-Value');
grid;
subplot(212),
plot(wt,amp2);
title('Approximate 2-order System Frequency Response');
xlabel('Frequency-rad');
ylabel('Amplitude-Value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“sysapprox.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysapprox，在 MATLAB Command Window 下查看运行的结果。

小结：本例使用 freqs ( ) 函数代替了前例的 polyval ( ) 函数，从效果上没有什么区别。当然，调用 freqs ( ) 函数显得更简洁一些。不过使用 polyval ( ) 函数是按照原始的频率特性函数表达式的定义来计算，更容易理解频率特性函数的本质。如果调用 freqs ( ) 函数不规定返回值，就会将系统频率特性函数的数值在 MATLAB Command Window 下显示出来。不过，如果希望计算某一特定频率下系统的频率响应数值，还是调用 polyval ( ) 函数更加方便一些。

得到系统的频率响应数据之后，除了可以按照直角坐标直接绘制系统的频率特性响应曲线之外，还可以根据不同需要选取半对数坐标或极坐标绘制系统的 Bode（波特）图和 Nyquist（奈奎斯特）图。

9.2.2 Bode 图绘制

如果把频率特性函数 G（jω）的角频率ω和幅频特性|G（jω）|都取对数，有：

$$\begin{cases}\mu = \lg\omega \\ L(G(\mathrm{j}\omega)) = \lg\left|G(\mathrm{j}\omega)\right| \\ \theta(G(\mathrm{j}\omega)) = \arg G(\mathrm{j}\omega)\end{cases}$$

这样得到的函数 L（G（jω））和θ（G（jω））分别称为对数幅频特性函数和对数相频特性函数，总称对数频率特性函数，注意θ不取对数。以μ为横坐标，L（G（jω））和θ（G（jω））为纵坐标绘制的曲线分别被称为对数幅频特性图和对数相频特性图，统称为系统的 Bode 图。

习惯上，不管|G（jω）|的量纲和单位是什么，统一给 L 规定一个名称，称为增益；又统一给 L 规定一种单位，称为贝尔（B）。同时规定 1 贝尔等于 20 分贝尔，简称分贝。此外给μ也规定一种单位，称为十倍频程。不过十倍频程只用来度量μ的两个值之差，而不用来度量μ值本身。

用 Bode 图表示对象的性质优点很多。首先，可以展宽视野。在对数坐标图上，系统的低频、中频和高频段的性质都可以以适当的比例清晰的展现出来。而在直角坐标图上，只有中频段表现的比较充分，低频段和高频段都不能得到充分反映。其次，曲线形状比较简单，易于手工绘制。这在缺乏计算机辅助设计软件的场合是非常重要的。第三，由于对数运算将四则运算的乘除化为加减，因此，可以用简单叠加的方式得到方框图串联形式连接的系统的总体的 Bode 图。

除了根据系统频率响应数据直接绘制 Bode 图外，MATLAB 还提供了一条函数 bode（）可以直接求解和绘制系统的 Bode 图。其基本调用格式为：

[MAG,PHASE] = BODE(SYS,...,W)

其中 SYS 是系统的频率特性函数或状态空间描述（MIMO 系统要指明是对第几个输入量）；W 是指定的角频率向量，也可以不加指定而由 MATLAB 自己给出；MAG 和 PHASE 是返回的系统开环幅频特性和相频特性，如果没有返回值的话 MATLAB 就在屏幕上绘制出缺省参数的系统 Bode 图。

MATLAB 还提供了一条函数 margin（），可以求解系统的增益裕量和相角裕量，其基本调用格式为：

[Gm,Pm,Wcg,Wcp] = MARGIN(SYS)

其中 Gm、Pm、Wcg 和 Wcm 分别是系统的增益裕量、相角裕量及其对应的角频率。

某控制系统的开环传递函数如下。试绘制其 Bode 图和增益裕量以及相角裕量，并与系统的闭环阶跃响应曲线比较。

$$G(s) = \frac{0.86}{s(0.36s+1)(0.3906s^2+0.75s+1)}$$

结果：系统的 Bode 图如图 9-10 所示。

图 9-10 的横坐标均为频率弧度值，注意其刻度是以 10 的幂的形式标注的。其中上半部分是系统的相频响应曲线，纵坐标为增益的 dB 值，下半部分为系统的幅频响应曲线，纵坐

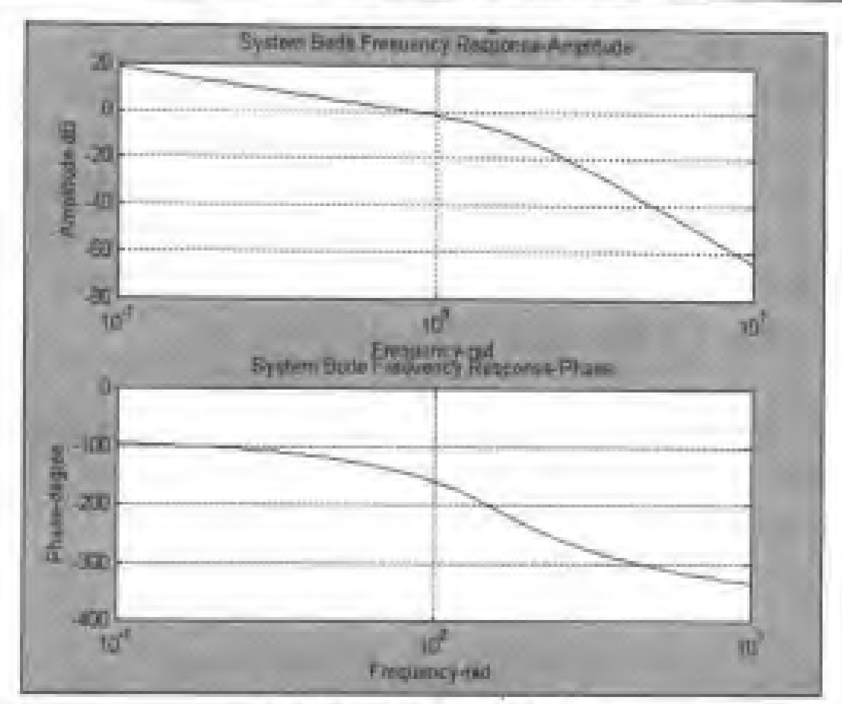


图 9-10　四阶系统 Bode 图

标为相角的角度值。

闭环系统阶跃响应曲线如图 9-11 所示。其横坐标为时间，纵坐标为系统的阶跃响应值。

系统的增益裕量和相角裕量为

System Gain Margin and its associated frequency are:

Gm =　1.5791

Wcg =　1.2304

System Phase Margin and its associated frequency are:

Pm = 30.8985

Wcm =　0.8561

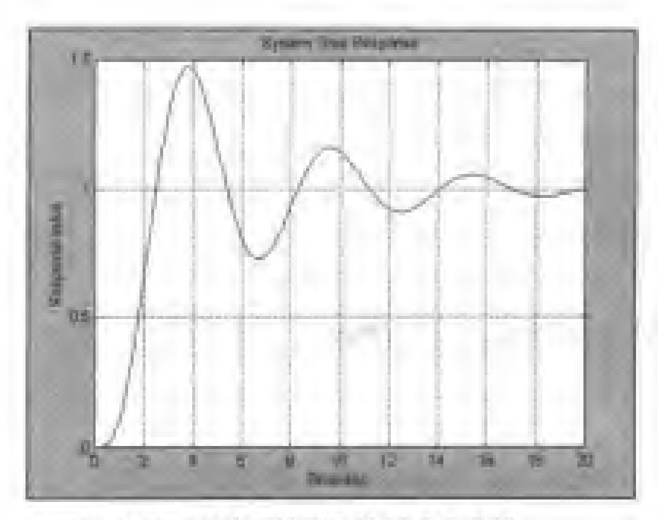


图 9-11　四阶系统闭环阶跃响应曲线

分析：从系统的 Bode 图上可以看出，低频段对数幅频特性曲线的斜率约为-20dB 每十倍频程，这是因为系统的开环传递函数有积分项 1/s。相应的对数相频特性曲线也是从-90° 开

始的，并且因为是四阶系统，最后趋于-360°。

系统的增益裕量约为 1.5791/dB，相角裕量约为 30.8985°，从系统 Bode 图的 0dB 线和-180°线也可以大致得到这个结果。从稳定裕量上看，系统是稳定的，系统的闭环阶跃响应曲线也可以验证这一点。不过系统的截止角频率较小，约 0.8561。因此从系统的闭环阶跃响应曲线上来看，系统的过渡过程时间较长（约 20s），超调量较大（约 50%），振荡次数也较多。

对于这样的高阶系统，仅仅用二阶系统近似一般是不能满足要求的，一般都要作串联校正或反馈校正，将系统的主导极点配置到合适的位置上来。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"sysbode.m"（注意不要直接取名为 bode.m，否则会冲掉 MATLAB 系统原有的 bode.m 文件）：

```
%关闭所有图形窗口并清除内存变量
close all;
clear all;
%系统开环传递函数描述
numo=[0 0 0 0 0.86];
den1=[1 0];
den2=[0.36,1];
den3=[0.3906 0.75 1];
deno=conv(den1,conv(den2,den3));
%系统闭环传递函数描述
numc=numo;
denc=deno+numo;
%系统仿真时间及频率初始化
t=0:0.1:20;
wt=logspace(-1,1);
%系统频率响应数据
[mag,phase]=bode(numo,deno,wt);
%系统增益裕量及相角裕量
[Gm,Pm,Wcg,Wcm]=margin(numo,deno);
%显示结果
disp('System Gain Margin and its associated frequency are:');
Gm
Wcg
disp('System Phase Margin and its associated frequency are:');
Pm
Wcm
%系统的闭环阶跃响应
y=step(numc,denc,t);
```

```
%图形绘制
subplot(211),
%对数幅频特性函数
amp=20*log10(mag);
semilogx(wt,amp)
title('System Bode Frequency Response-Amplitude');
xlabel('Frequency-rad');
ylabel('Amplitude-dB');
grid;
subplot(212),
semilogx(wt,phase);
title('System Bode Frequency Response-Phase');
xlabel('Frequency-rad');
ylabel('Phase-degree');
grid;
plot(t,y);
title('System Time Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“sysbode.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysbode，在 MATLAB Command Window 下查看运行的结果。

小结：本例用到了一个新的 MATLAB 函数 conv ( )，其功能是求解两个向量的卷积或求解两个多项式的乘积。熟悉信号处理的读者都清楚，这两者在数学上是一致的。其基本调用格式为 C = CONV(A, B)，其中 A 和 B 是两个向量，可以代表待卷积的序列，也可以代表多项式的系数（不必限制一定是降幂排列，但两者的排列方式必须一致）；C 是返回的向量，维数是 LENGTH(A)+LENGTH(B)-1。对于调用该函数求解多项式乘积来说，需要注意的是，此函数只能计算两个多项式的乘积。如果希望计算 n 个多项式的乘积，或者调用 conv ( ) 函数 n-1 次，或者自己编写相应的 M 函数。

本例在绘制 Bode 图和求解稳定裕量时用的是开环频率特性函数，而绘制阶跃响应曲线时用的是闭环频率特性函数。这是因为最初系统的闭环频率特性图是难于绘制的，一般是通过开环频率特性去估计。MATLAB 继承了这一做法，因此 bode ( ) 函数和 margin ( ) 函数都是为开环频率特性函数编写的。当然读者也可以编写求解闭环频率特性函数 Bode 图和稳定裕量的函数。

控制理论中还有一种对数频率响应图称为对数幅相特性图。其含义是在一定的频率范围内，以相角 θ 作为横坐标，以对数幅值 L 作为纵坐标，以角频率 ω 作为参变量绘制的频率特性图，也称为 Nichols 图。MATLAB 同样提供了绘制此对数幅相特性图的函数 nichols ( )，其调用格式与 bode ( ) 函数完全一致。例如执行完前例的 sysbode.m 文件之后，在 MATLAB

Command Window 下键入

```
nichols(numo,deno)
```

可得该系统的对数幅相特性曲线如图 9-12 所示。其横坐标为开环系统的相角，单位是度；纵坐标为系统的开环增益，单位是分贝(dB)。

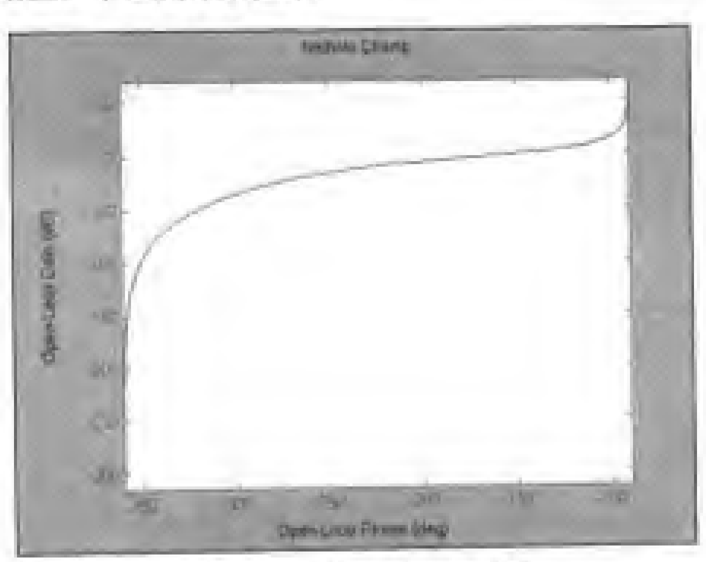


图 9-12　系统的 Nichols 曲线

从系统的对数幅相特性曲线中同样可以读出系统的增益裕量和相角裕量，不过相应的角频率 ω 就需要重新计算了。

9.2.3　Nyquist 图绘制

按照频率特性函数的本意，频率 ω 是正实数，但是如果不管物理意义，只把它看作是 ω 的一个函数，那么我们也可以考察当 ω 为负数时的频率特性函数的图像。一切集总参数对象的传递函数 $G(s)$ 都是 s 的有理函数，所以它们的频率特性函数都是 $\mathrm{j}\omega$ 的有理函数。因此容易理解，对于任一给定的频率 ω，$G(\mathrm{j}\omega)$ 和 $G(-\mathrm{j}\omega)$ 必定是互为共轭的一对复数，它们在复平面上的位置是关于实轴对称的。

习惯上，我们称频率特性函数 $G(\mathrm{j}\omega)$ 在复平面的图像为系统的极坐标频率特性图或 Nyquist 图。

为了方便，有时要研究 $G(\mathrm{j}\omega)$ 的倒数，称为逆频率特性函数，定义为

$$\hat{G}(\mathrm{j}\omega)=\frac{1}{G(\mathrm{j}\omega)}$$

显然，逆幅频和逆相频特性函数与幅频和相频特性函数有如下的关系：

$$\begin{cases}\left|\hat{G}(\mathrm{j}\omega)\right|=\dfrac{1}{|G(\mathrm{j}\omega)|}\\ \arg\hat{G}(\mathrm{j}\omega)=-\arg G(\mathrm{j}\omega)\end{cases}$$

它们的逆幅频和逆相频特性函数图像就称为逆 Nyquist 图，在多变量频域设计的对角化方面有一定的应用。

通过系统的开环频率特性函数 Nyquist 图，还可以判断系统的稳定性，这就是著名的

Nyquist 稳定判据。Nyquist 稳定判据叙述如下：

如果系统的开环传递函数 $G(s)$在右半 s 平面有 p 个极点，则闭环系统稳定的充分必要条件是：当 ω从$-\infty$连续的变化到$+\infty$时，开环频率特性函数 $G(j\omega)$的 Nyquist 图逆时针方向包围复平面上的-1 点 p 圈。

应用 Nyquist 稳定判据研究控制系统的稳定性有很多优点：首先，它主要靠作图，手工计算量很小。其次，它不仅能回答闭环系统是否稳定，而且还可以得到系统接近于不稳定的程度。更进一步，这个判据往往还可以提示改善系统稳定性的办法。除此之外，在实际应用中这个判据可以不要求知道系统的微分方程或传递函数，而只要依靠实验测出其开环频率特性即可应用。

MATLAB 控制系统工具箱中提供了一条函数 nyquist（），可以用来求解或绘制系统的 Nyquist 图。其基本调用格式为

[RE,IM] = NYQUIST(SYS,...,W)

其中 SYS 是系统的频率特性函数或状态空间描述（MIMO 系统要指明是对第几个输入量）；W 是指定的角频率向量，也可以不加指定而由 MATLAB 自己给出；RE 和 IM 是返回的系统开环频率特性值的实部和虚部，如果没有返回值的话，MATLAB 就在屏幕上绘制出缺省参数的系统 Nyquist 图。请看下例：

某控制系统的开环传递函数如下页所示。试绘制当 K=2 和 K=20 时系统的 Nyquist 图。根据 Nyquist 稳定判据判断系统的稳定性，并绘制系统的闭环阶跃响应曲线验证关于稳定性的结论。

$$G(s)=\frac{K}{s(s+1)(0.1s+1)}$$

结果：系统在 $K=2$ 时的 Nyquist 图如图 9-13 所示。其横坐标为系统频率响应的实部，纵坐标为虚部。

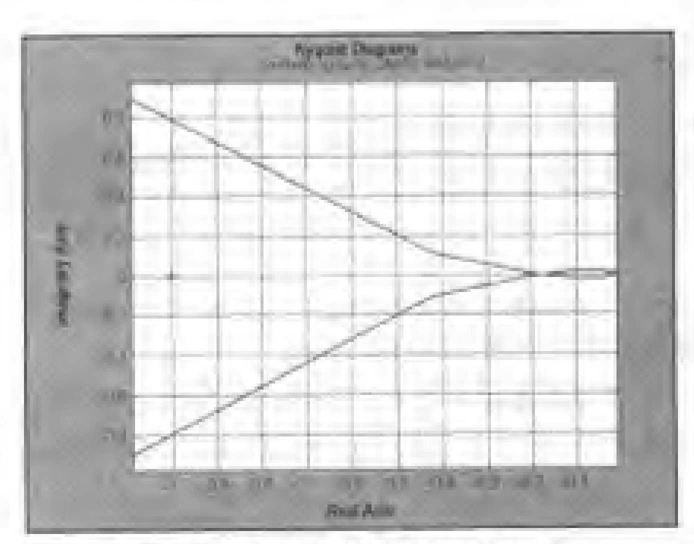


图 9-13 K=2 时系统的 Nyquist 曲线

$K=20$ 时系统的 Nyquist 曲线如图 9-14 所示。同样，其横坐标为系统频率响应的实部，纵坐标为虚部。

K=2 和 K=20 系统的闭环阶跃响应曲线如图 9-15 所示。

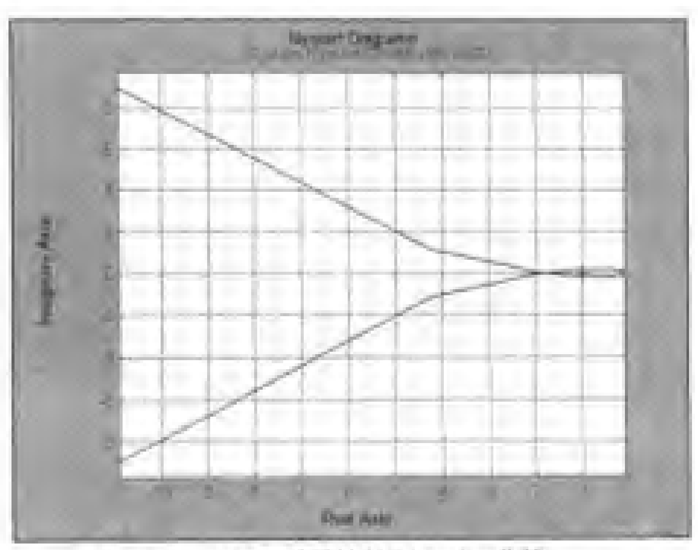


图 9-14　K=20 时系统的 Nyquist 曲线

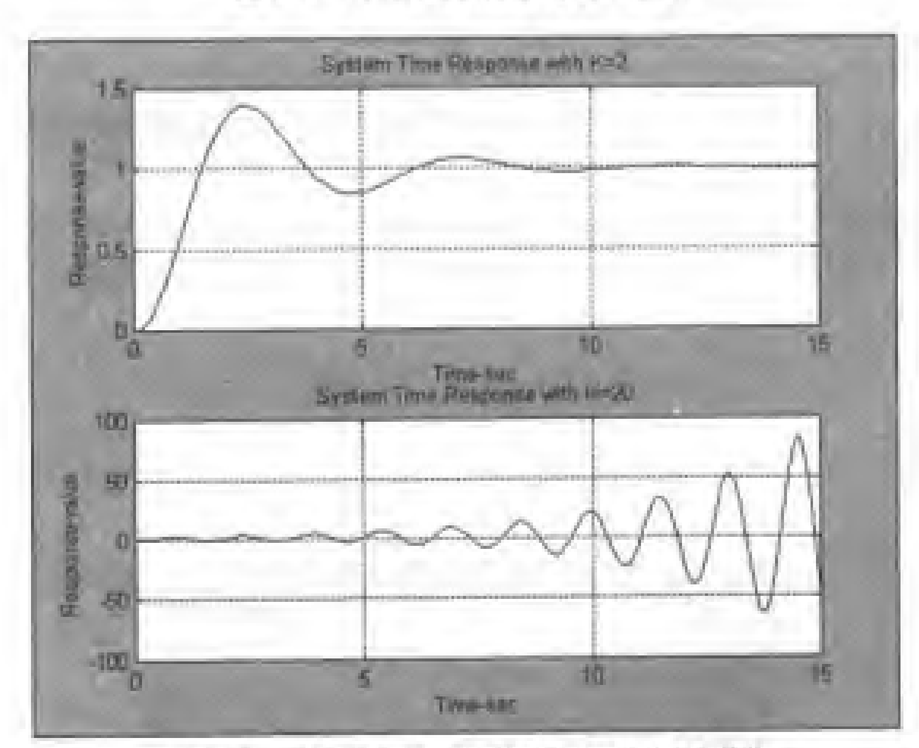


图 9-15　K=2 和 K=20 时系统的闭环阶跃响应曲线

其中上半部分是 K=2 时系统的阶跃响应曲线，下半部分是 K=20 时系统的阶跃响应曲线。其横坐标均为时间，纵坐标均为系统的阶跃响应值。

分析：根据系统的开环传递函数表达式，系统在复平面右半平面的极点数为 0。因此，该系统稳定的充分必要条件为：系统开环频率特性函数 $G(\mathrm{j}\,\omega)$的 Nyquist 曲线不包含复平面的(−1，0)点。

从图 9-13 的 $K=2$ 时系统 Nyquist 图来看，图中用“+”号表示的复平面的(−1，0)点远离系统的 Nyquist 曲线，因此，系统的闭环时间域响应是稳定的。图 9-15 上半部分 $K=2$ 时系统的闭环阶跃响应曲线也验证了这种说法，系统的输出量在振荡两次后大约从 10s 开始进入稳态值 5%左右的区间内，并最终趋于设定值 1。

从图 9-14 的 $K=20$ 时系统 Nyquist 图来看，图中用“+”号表示的复平面的(−1，0)点被

系统的 Nyquist 曲线包围。当 ω 从-∞连续的变化到+∞时，“+”号被系统的 Nyquist 曲线包围两圈，说明系统在复平面的右半平面有两个极点。因此，系统的闭环时间域响应是不稳定的。图 9-15 下半部分 $K=20$ 时系统的闭环阶跃响应曲线同样验证了这种说法，系统的输出量增幅振荡，最终趋于-∞。

注意：这里用的是“验证”而非“证明”，因为直观的定性观察并不能作为理论推导的结果。因为本例系统的开环频率特性函数是 jω 的有理函数，所以实轴以上的部分应该与实轴以下的部分关于实轴对称，图 9-13 和图 9-14 的系统 Nyquist 曲线都验证了这种说法。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件“sysnyquist.m”（注意不要直接取名为 nyquist.m，否则会冲掉 MATLAB 系统原有的 nyquist.m 文件）：

```
%关闭所有图像窗口并清除内存变量
close all;
clear all;
%K=2 时系统的开环频率特性函数
numo1=[0 0 0 2];
den1=[1 0];
den2=[1 1];
den3=[0.1 1];
deno1=conv(den1,conv(den2,den3));
%K=20 时系统的开环频率特性函数
numo2=[0 0 0 20];
deno2=deno1;
%系统的闭环频率特性函数
numc1=numo1;
denc1=deno1+numo1;
numc2=numo2;
denc2=deno2+numo2;
%仿真时间及角频率初始化
t=0:0.1:15;
wt=-10:1:10;
%K=2 时系统的开环 Nyquist 曲线
nyquist(numo1,deno1,wt);
title('System Nyquist Charts with K=2')
grid;
%K=20 时系统的开环 Nyquist 曲线
nyquist(numo2,deno2,wt);
title('System Nyquist Charts with K=20')
grid;
```

```
%系统的闭环阶跃响应
y1=step(numc1,denc1,t);
y2=step(numc2,denc2,t);
%绘图
subplot(211),
plot(t,y1);
title('System Time Response with K=2');
xlabel('Time-sec');
ylabel('Response-value');
grid;
subplot(212),
plot(t,y2);
title('System Time Response with K=20');
xlabel('Time-sec');
ylabel('Response-value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“sysnyquist.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysnyquist，在 MATLAB Command Window 下查看运行的结果。

小结：使用 nyquist ( ) 函数绘制系统的开环 Nyquist 曲线时，尤其要注意函数的调用格式和参数设置。因为应用 Nyquist 稳定判据时关键就在系统 nyquist 曲线在复平面 (-1，0) 点附近的情况，所以在选择参变量 ω 时就既要能反映曲线在 (-1，0) 点附近的变化情况，又要能看出曲线在 ω 趋于 ∞ 时的变化趋势。因此，对于不同的系统 ω 的选择是不同的。当然，使用系统给定的 ω 也是一种办法，但往往不能奏效。另外，如果使用 nyquist ( ) 而非 plot ( ) 函数绘图，MATLAB 自动在图中 (-1，0) 点标出一个“+”号，方便使用者应用 Nyquist 稳定判据。

9.2.4 离散系统的频率响应

离散控制系统又称采样控制系统，是一种采用断续方式控制的系统。一般这种系统中都含有通常称为采样器的专门开关装置，定时开启和关闭。近几十年由于数字式计算机的飞速发展，离散控制系统在现代工业控制中获得了非常广泛的应用。数字控制器或计算机在控制速度、控制精度以及性能价格比方面都比模拟控制器有着明显的优越性，并且数字控制器还有量化的通用性。事实上，不管是连续系统还是离散系统，使用数字式计算机进行仿真时都要化成离散时间系统。

离散控制系统的理论基础是采样定理，数学工具是 Z 变换。相信读者都已经在相关的课程中有所接触，这里就不再赘述了。

在 MATLAB 环境下离散控制系统时间域仿真的函数 dstep ( ) 和 dimpulse ( )，频率域的情形也基本类似，同样是连续系统的频率响应函数前边加 d 就成了相应的离散系统的频率域分析函数。例如 dbode ( ) 函数、dnyquist ( ) 函数和 dnichols ( ) 函数，分别是求取离散系

统的 Bode 图、Nyquist 图和 Nichols 图。这些函数的调用格式和参数设置和相应的连续系统函数也大体相同，不过需要注意的是，这些函数的参数中多了采样时间 Ts 这样一个必选项，其功能是设定该离散相同仿真的时间间隔。下面以一个简单的例子使读者熟悉一下这些函数的用法。

某控制系统传递函数如下，在其输入部分加一个 Ts=2s 的采样器，试求其离散系统模型并绘制系统的 Bode 图、Nyquist 图和 Nichols 图。

$$G(s)=\frac{2}{10s+1}\cdot\frac{1.2}{2s+1}$$

结果：离散系统的 Bode 图如图 9-16 所示。

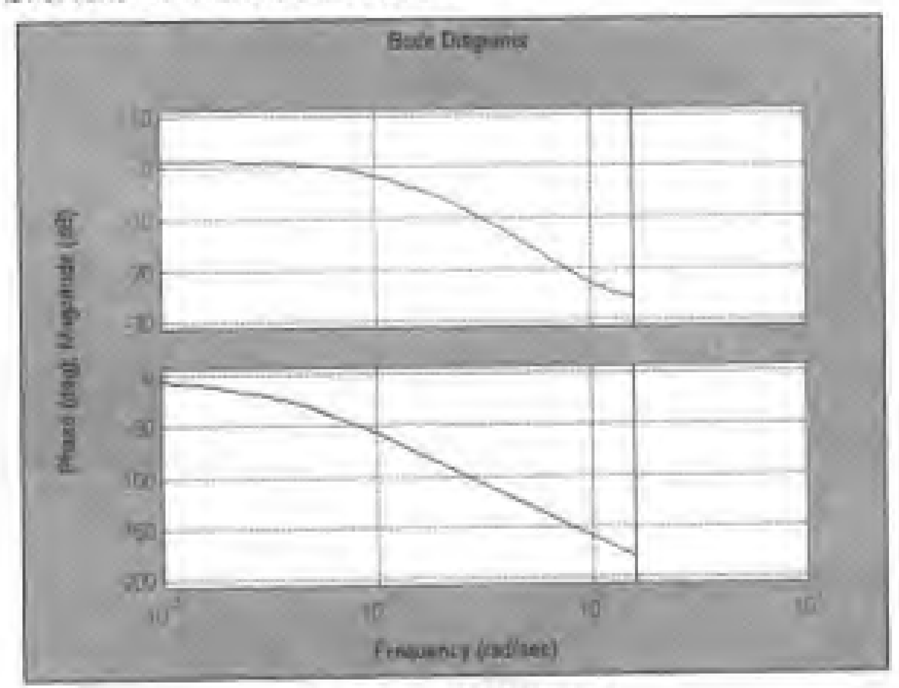


图 9-16　离散系统 Bode 图

其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。横坐标均为频率，单位为 rad/s；纵坐标分别是增益，单位为分贝(dB)和相角，单位为度(°)。

离散系统的 Nyquist 图和 Nichols 图见图 9-17。其中上半部分是离散系统的 Nyquist 图，横坐标是系统频率响应实部，纵坐标为频率响应的虚部；下半部分是离散系统的 Nichols 图，横坐标为开环系统的相角，单位是度（°）；纵坐标为系统的开环增益，单位是分贝(dB)。

离散系统模型为

$$H(z)=\frac{0.135z}{(z-0.819)(z-0.368)}$$

求解过程：

本例的解题步骤分为两部分。

1．求解离散系统的脉冲传递函数

由：

$$G(s)=\frac{2}{10s+1}\cdot\frac{1.2}{2s+1}=\frac{3}{10s+1}-\frac{0.6}{2s+1}$$

取拉普拉斯逆变换，然后在取 Z 变换，并代入 Ts=2，得：

$$H(z)=\frac{0.3z}{z-e^{-0.1T}}-\frac{0.3z}{z-e^{-0.5T}}=\frac{0.135z}{(z-0.819)(z-0.368)}$$

上述过程可以用前边介绍过的 residue（）函数得到。

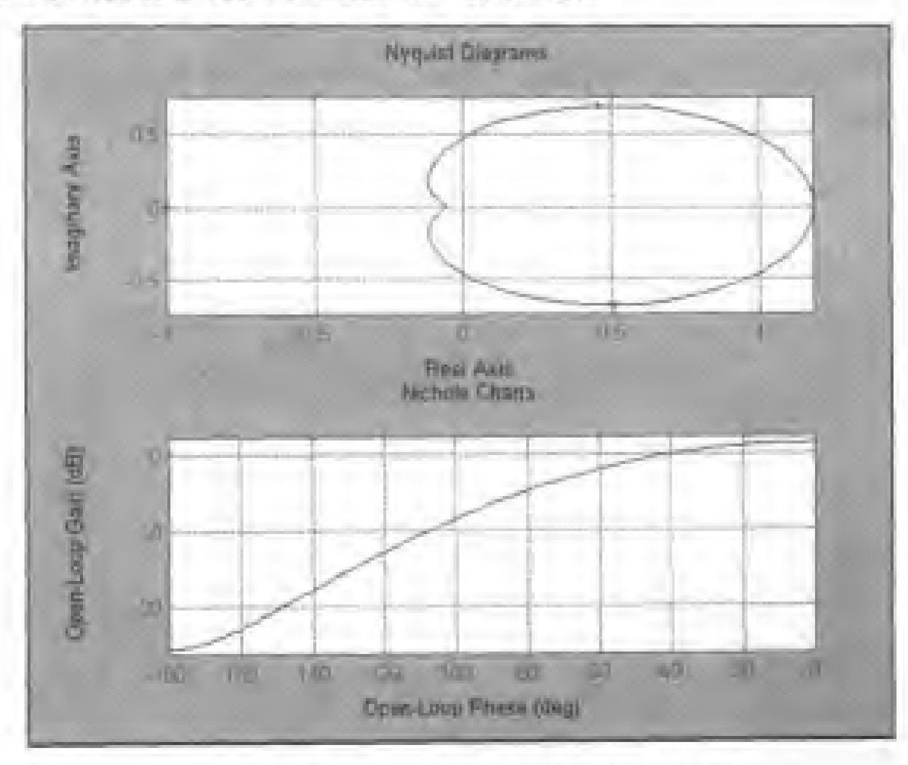


图 9-17　离散系统的 Nyquist 图和 Nichols 图

2. 编写求解频率响应曲线的 M 文件

因为本例只是示意函数的用法和效果，并非根据曲线分析系统，所以程序编写得比较简单。可以在 MATLAB Editor/Debugger 下编辑存盘，也可以在 MATLAB Command Window 下直接键入执行：

```
%系统的开环模型
numo=[0 0.135 0];
den1=[1 -0.819];
den2=[1 -0.368];
deno=conv(den1,den2);
%采样时间
Ts=2;
%系统 Bode 图、Nyquist 图和 Nichols 图绘制
dbode(numo,deno,Ts);
grid;
subplot(211),
dnyquist(numo,deno,Ts);
grid;
subplot(212),
```

```
dnichols(numo,deno,Ts);
grid;
```

小结：从本例可以看出，调用离散系统的频率响应函数和调用连续系统的相应函数没什么太大区别，只不过注意一下采样时间 Ts 而已。当然，同样的连续系统采用不同的采样时间，得到的可能是完全不同的离散系统。并且有可能把稳定的系统变为不稳定的 因此，在进行离散系统分析时，尤其要注意采样时间的选择。

9.3 根轨迹的绘制

根轨迹法是 W.B.Evans 于 1948 年提出的一种求解变换特征方程根的简便的图解方法，在工程上获得了广泛的应用。它根据系统开环传递函数极点和零点的分布，依照一些简单的规则，用作图的方法求出闭环极点的分布，避免了复杂的数学计算。有些读者可能会问，既然使用 MATLAB 控制系统工具箱提供的函数可以很方便的求出系统的闭环传递函数及其零点、极点和增益，那么类似根轨迹法这种间接分析系统闭环传递函数的方法不就没有生命力了吗？其实不是这样的。一方面，应用根轨迹法可以很容易地从图上看出来系统中某个参数的变化会对系统的闭环极点产生什么影响，进而如何影响系统的动态性能；另一方面，完整的根轨迹图形也是控制系统设计和校正的一种得力的辅助工具和检验手段。另外，应用根轨迹法的手工计算量很小，结果也比较可靠，在没有控制系统计算机辅助设计软件的情况下未尝不是一种很好的选择。下边简要介绍一下根轨迹的简便特性及绘制规则。

所谓根轨迹，就是指系统的开环传递函数增益变化时其变化传递函数的极点在复平面上变化的轨迹。

设系统的开环传递函数为：

$$G_o(s)=\frac{KN(s)}{D(s)}$$

其中 N（s）和 D（s）分布是 s 的 m 次和 n 次多项式，且 n≥m。则闭环特征方程为：

$$KN(s)+D(s)=0$$

显然，当 N（s）和 D（s）没有公因子时，这个方程与下面的方程 $G_0(s)=-1$ 同根。

因此，可以将系统的开环传递函数写成下述形式：

$$G_o(s)=\frac{KN(s)}{D(s)}=\frac{K(s-z_1)\cdots(s-z_m)}{(s-p_1)\cdots(s-p_n)}$$

将上式两端分别取模和取角，就得到绘制根轨迹的模条件和角条件分别为：

$$K\frac{\prod_{i=1}^{m}|s-z_i|}{\prod_{j=1}^{n}|s-p_j|}=1$$

$$\sum_{i=1}^{n}\arg(s-z_i)-\sum_{j=1}^{n}\arg(s-p_j)=(2k+1)\pi$$

以上就是绘制闭环传递函数根轨迹的理论基础。人们根据绘制根轨迹的模条件和角条件

总结了以下几条根轨迹绘制的基本规则：

(1) 根轨迹的对称性：根轨迹关于实轴对称。

(2) 根轨迹的分支数、起点和终点：对 n>m，根轨迹共有 n 条分支。根轨迹的起点是开环极点，有 m 条根轨迹的终点是开环零点，其余 n-m 条根轨迹的终点在无穷远点，称为无穷远开环零点。

(3) 根轨迹在实轴上的分布：实轴上的某一段属于系统闭环传递函数的根轨迹，当且仅当其右侧开环传递函数的实极点和实零点数目之和为奇数。

(4) 根轨迹的渐近线：根轨迹共有 n-m 条渐近线，都是从实轴上的共同交点向外辐射，辐射角（亦称渐近线的倾角）为：

$$\theta=\frac{2k+1}{n-m}\pi$$

(5) 根轨迹的分离点及会合点：如果实轴上相邻两极点间的线段属于根轨迹，则它们之间必有分离点。同理，如果实轴上相邻两零点间的线段属于根轨迹，则它们之间必有会合点。

(6) 实轴上分离点的分离角及会合点的会合角：实轴上分离点的分离角恒为±90°。同理，实轴上会合点的会合角恒为±90°。

(7) 根轨迹在复极点（或复零点）处的出射角（或入射角）：出射角就是从复极点 p 出发的根轨迹切线的方向角，其值为：出射角＝Σ[各零点指向本极点方向的方向角]－Σ[其他极点指向本极点的方向角]＋反向。

(8) 根轨迹与虚轴的交点及临界根轨迹的增益值：将 s=jω 代入特征方程可求得 ω 和 K，即根轨迹与虚轴交点的坐标及交点所对应的临界根轨迹增益值。

通过以上这些规则，一般都可以绘制出比较满意的根轨迹示意图。当然，MATLAB 提供了一条函数 rlocus ()，可以通过系统的开环传递函数绘制其闭环的根轨迹图。不必再手工绘制。其基本调用格式为

R = RLOCUS(SYS,K)

其中 SYS 是系统的开环传递函数描述，K 是系统增益向量，也可以缺省并使用 MATLAB 提供的向量；R 是返回的根轨迹数据。如果不设返回值，MATLAB 就自动绘制系统的闭环传递函数的根轨迹。请看下例：

某控制系统的开环传递函数如下，试绘制系统的闭环根轨迹。寻找系统临界稳定时的增益 K，并绘制此时的系统阶跃响应作为验证。

$$G_o(s)=\frac{K}{(s+1)(s+3-\mathrm{j}1)(s+3+\mathrm{j}1)}$$

结果：系统的闭环传递函数根轨迹如图 9-18 所示。

其横纵坐标轴在图中以蓝色实线描出，分别是系统频率响应的实部和虚部。图中带间断点的粗实线就是系统的根轨迹。

系统的临界稳定增益为：

System critical stable-point is:

K(i)=102

临界稳定时系统的阶跃响应曲线如图 9-19 所示。其中横坐标为时间，纵坐标为系统的响应值。

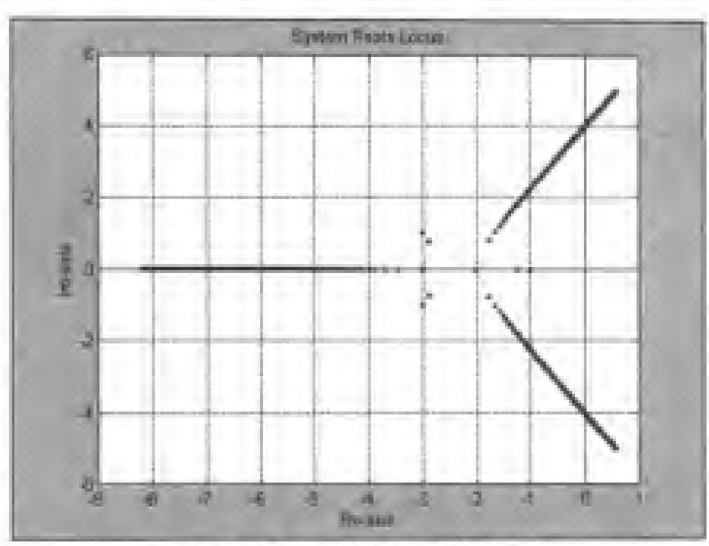


图 9-18 系统闭环根轨迹图

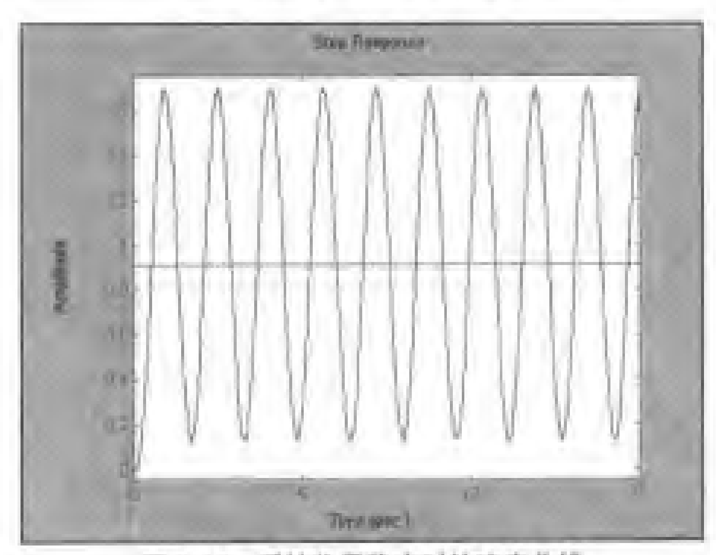


图 9-19 系统临界稳定时的响应曲线

分析：从图 9-18 中可以看出，系统的闭环根轨迹有 3 条分支，对应系统的极点与零点之差也为 3。出射角方向为 180° ÷3＝60° 。系统的临界稳定增益为 102，此时系统的阶跃响应为等幅振荡（见图 9-19）。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件 "sysrlocus.m"（注意不要直接取名为 rlocus.m，否则会冲掉 MATLAB 系统原有的 rlocus.m 文件）：

```
%系统开环传递函数
num=1;
den1=[1 1];
den2=[1 3-sqrt(-1)];
den3=[1 3+sqrt(-1)];
den=conv(den1,conv(den2,den3));
%增益相邻初始化
```

```
K=0:1:200;
%求解根轨迹数据
r=rlocus(num,den,K);
%寻找临界稳定点
[m,n]=size(r);
for i=1:m
    if real(r(i,2))>0
        break;
    end
end
%显示结果
disp('System critical stable-point is:')
K(i)
%根轨迹绘图
x=-9:0.1:1;
y=-6:0.1:6;
tx=zeros(1,length(x));
ty=zeros(1,length(y));
plot(r,'.');
line(x,tx);
line(ty,y);
title('System Roots Locus');
xlabel('Re-axis');
ylabel('Im-axis');
grid;
%临界稳定点系统闭环传递函数
numc=zeros(1,length(den));
numc(length(den))=K(i);
denc=den+numc;
%临界稳定时系统的阶跃响应曲线
step(numc,denc);
```

选择“File”菜单的“Save”选项，保存文件名为“sysrlocus.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysrlocus，在 MATLAB Command Window 下查看运行的结果。

小结：本例使用了一个新函数 line ( )，其功能是在指定的位置画直线。

在本章中我们主要介绍了 MATLAB 环境下控制系统时域响应分析、频率响应分析，包括 Bode 图绘制、Nyquist 图绘制、离散控制系统频率响应，还有根轨迹的绘制等等。对于以上各部分内容，其关键有两点：一是求解系统的响应曲线，对系统的性能作出评价；二是根据各种间接的方法判断系统的稳定性。

第 10 章 传递函数模型控制系统校正

在前面几章里，我们讨论了控制系统的两种工程分析方法：时间域方法和频率域方法（根轨迹方法也是一种频率方法）。利用这些方法我们能够在系统结构和参数已确定的条件下，计算和估计它们的性能。这个问题通常被称作系统的分析问题。但在工程实际中常常提出相反的要求，就是说，被控对象是已知的，性能指标是预先给定的，要求设计者选择合适的结构和参数，使控制器与被控对象组成一个性能满足要求的系统。这个问题叫做系统的综合。可见，综合的目的就是在系统中引入合适的附加装置，使原有系统的缺点得到校正，从而满足一定的性能指标。引入的附加装置通常称为校正装置。所以系统的综合问题就是选择校正装置接入的位置以及它的结构参数的问题。有时也笼统的把系统的综合称为系统的校正。本章所要介绍的，主要是 MATLAB 在单变量控制系统校正中的一些应用，其重点的函数都包含在控制系统工具箱中。

对于单变量系统来说，校正装置接入系统的主要形式有两种：一种是校正装置与被校正对象相串联，如图 10-1（a）所示，这种校正方式称为串联校正。另一种形式是从被校正对象中引出反馈信号，与被校正对象或其一部分构成局部反馈回路，并在局部反馈回路内设置校正装置。这种校正方式称为局部反馈校正或“并联校正”，如图 10-1（b）所示。本章将重点讲述有关串联校正的内容。

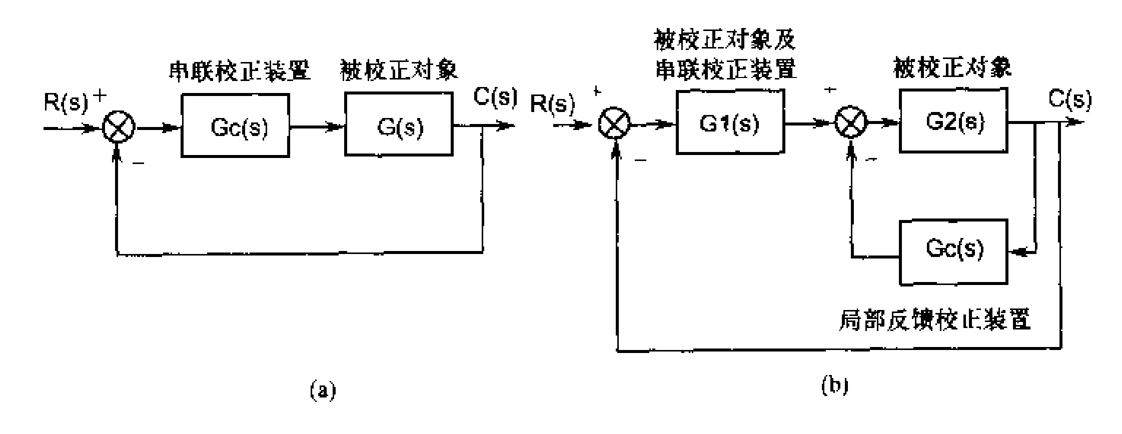


图 10-1 控制系统校正的两种主要形式

MATLAB 控制系统工具箱中提供了一些用于串联校正和局部反馈校正的函数。不过更重要的是，MATLAB 提供了一个方便的作图和数值计算环境，在这个环境下可以非常轻松地将

校正的指标和结果以图形的形式表示出来，结论一目了然。无论是理论推导还是工程应用，都有很强的辅助作用。

10.1 控制系统校正指标和经验公式

对于一个稳定的控制系统的要求通常可以归结为要求其输出量尽可能与输入量保持某种给定的关系（包括与输入量相等这种最简单的关系）。但是任何一个实际的控制系统都不能完全做到这一点，总存在着一定的误差。造成系统误差的原因很多，其中由于系统本身的结构和参数造成的误差称为原理性误差。此外，组成系统的元部件的不良特性，如摩擦、死区、间隙等非线性因素也都会产生误差。误差有静态的，也有动态的。从某种意义上说，误差的大小就可以表明一个控制系统性能的优劣。从这一点出发，工程上形成了一些指标，用以衡量控制系统的性能，这些性能指标尽管提法不同，但都体现了对系统静态特性和动态特性要求。

性能指标要反映系统实际性能的特点，又要便于量测和检验。所以对于不同类型的系统，对于不同的研究和应用领域，采用不同的性能指标。性能指标的提法有很多种，大体上可以归纳为两类：时间域指标和频率域指标。这两类指标在前面各章中都曾经简单介绍过，现在以图 10-2 MATLAB 绘制的二阶系统响应曲线为例再做一简单回顾。

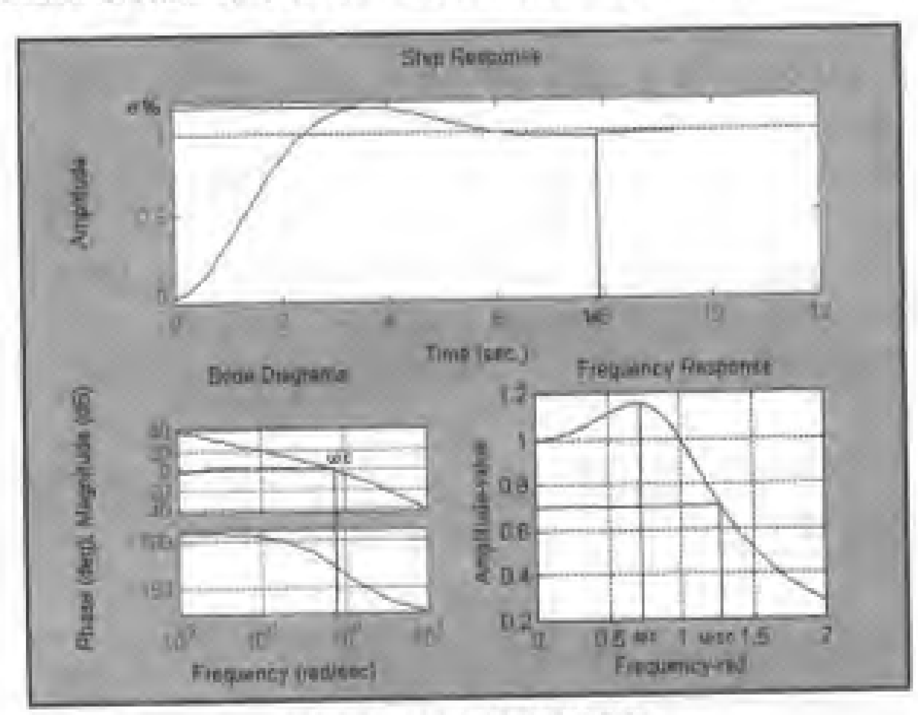


图 10-2 控制系统的性能指标

其中上半部分是系统的阶跃响应，横坐标是时间，纵坐标是响应值；左下部分是系统的 Bode 图，横坐标为频率，单位 rad/s，纵坐标分别是增益 dB 值和相角（°）；右下部分是未取对数的系统频率响应曲线。

时间域指标包括静态指标与动态指标。

(1) 静态指标：指的是静态误差 e、无静差度，以及开环比例系数。这里 e 是指系统在跟



踪典型输入（单位阶跃输入，单位斜坡输入和抛物线输入）时的静态误差。关于干扰和负载的变化所造成的静态误差也可以规定类似的指标。

(2) 动态指标：主要指的是过渡过程时间 ts 和超调量 $\sigma\%$（见图 10-2 上半部分系统的阶跃响应）。此外，上升时间、延迟时间、峰值时间、振荡次数等也都属于时间域动态指标。但为了简单起见，通常只用过渡过程时间和超调量来表示系统的动态特性。

频率域指标包括开环频率指标和闭环频率指标。

(1) 开环频率指标：指的是截止角频率 ω_c，相角稳定裕量 γ、增益稳定裕量 Kg 等。其中比较常用的是前两项。

(2) 闭环频率指标：主要指闭环谐振峰值 M_r 和谐振角频率 ω_r、闭环截止角频率 ω_{cc}（相当于闭环增益为 0.707（-3dB）时的角频率）。

无论是时间域指标还是频率域指标，都在一定程度上反映了系统的稳定性、快速性和准确性，只是反映的角度有所不同，所以各种指标之间必定存在着一定的联系。

不过还应看到，性能指标只是从某些工程角度大体上描述系统性能，有可能数学性质差别很大的系统，如不同阶次的系统，具有几乎相同的性能指标。因此不能指望在不同指标提法之间建立准确的数据对应关系。

但是在工程应用中，人们还是总结了一些经验公式来定性或定量地描述这些性能指标之间的关系。这些经验公式一般都是在时间中被证明是行之有效的，我们后边应用 MATLAB 对控制系统进行校正时也是以这些经验公式为基础的。不过如前所述，这些公式仅仅是经验性的，不具有同样的意义。

二阶系统的性能指标有着确定的关系，这里就不讨论了。对于高阶系统来说，在初步设计时可假设其谐振峰值 M_r 与相角裕量 γ、闭环截止角频率与开环截止角频率之间有如下的简洁关系：

$$M_r \approx \frac{1}{\sin\gamma}, \quad \omega_{cc} \approx \omega_c$$

应当指出，不同性能指标并不是完全等价的，M_r 和 γ 虽然都在一定程度上反映了系统的稳定性，但 M_r 包含的信息要比 γ 多。γ 只反映某一特定频率下系统的性质，而 M_r 则反映了一定频率范围内系统的性质。M_r 值大的系统，一般可以说振荡剧烈（稳定性差），但 γ 值大的系统，却不能保证一定有足够的稳定性。为了保证系统有足够的稳定裕量，一般希望 $M_r<1.4$，$\gamma>30°$，ω_c 的选择由系统的快速性要求来确定。以上述两式为基础，我们可以推导出一系列控制系统校正的经验公式：

(1) $M_r \approx \dfrac{1}{\sin\gamma}$，$\sigma(\%) = \dfrac{2000}{\gamma} - 20$

(2) $\sigma(\%) = \begin{cases} 100(M_r - 1), & M_r \leqslant 1.25 \\ 50\sqrt{M_r - 1}, & M_r < 1.25 \end{cases}$　　$\sigma = 0.16 + 0.4(M_r - 1)$

$$t_s = \frac{\pi}{\omega_c}[2 + (1.5M_r - 1) + 2.5(M_r - 1)^2]$$

$$M_r = \frac{h+1}{h-1}, \quad 1.1 \leqslant M_r \leqslant 1.8$$

(3) $t_s=(4\sim9)\omega_c$

其中 h 表示对数幅频特性图中频段斜率为 −1 的那一段宽度。为了简便起见，有时也把 h 称为中频段宽度。

10.2 系统开环频率特性设计

有了 MATLAB 这个得力的工具，系统的开环频率设计变得非常简便易行。我们只需选用几个简单的经验公式，不断地用 MATLAB 绘制的图形来校正控制器的设计，不必进行大量的计算，就可以得到满意的结果。本章的图形比较多，因为不是具体讲述图形绘制，而且空间有限，所以尽量把曲线压缩在比较小的范围内，以能看清刻度和关键值为宜。本章实例的讲述格式也与前几章有所不同，以边求解边显示结果为主，主要是为了显示解题的思路。并且，考虑到读者对 MATLAB 已经比较熟悉，有关建立 M 文件和存盘等步骤就不再详细叙述了。请看下例：

考虑如图 10-3 所示的电机定位系统，其系统传递函数如下页所示。相关参数为：

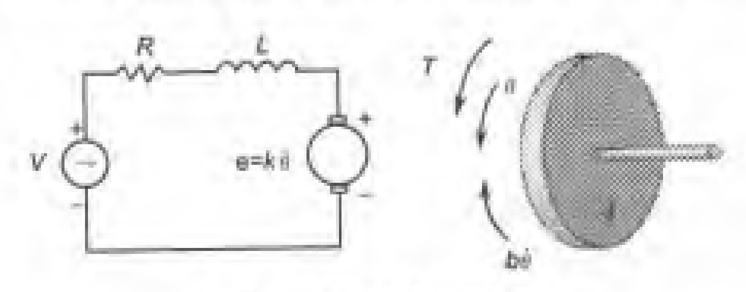


图 10-3 电机定位系统示意图

电机转动惯量 J= 3.2284E−6 kg·m²/s²，系统机械阻尼系数 b = 3.5077E−6 Nms，机电常量 $K=Ke=Kt$ = 0.0274 Nm/A，电阻 R= 4Ω，电感 L = 2.75E−6 H。输入量是电源电压 V，输出量是转轴的角度 θ，并假设系统的转轴是刚性的。试求解一校正装置，使系统的阶跃响应满足：

$$\frac{\theta(s)}{V(s)}=\frac{K}{s((Js+b)(Ls+R)+K^2)}$$

(1) 过渡过程时间小于 40ms；

(2) 超调量小于 16%；

(3) 静态误差为零；

(4) 扰动静态误差为零。

结果：求得的校正装置为：

$$G(s)=\frac{0.0182(s+60)(s+52.9)}{s}$$

校正后系统的阶跃响应如图 10-4 所示。其横坐标是时间，纵坐标是系统的阶跃响应值。

分析：校正装置是积分器加一个二阶的导前环节，功能是消除稳态误差，增强系统的快速性。从图 10-4 来看，系统的过渡过程时间小于 40ms，超调量小于 20%，并且稳态无差，基本满足要求。

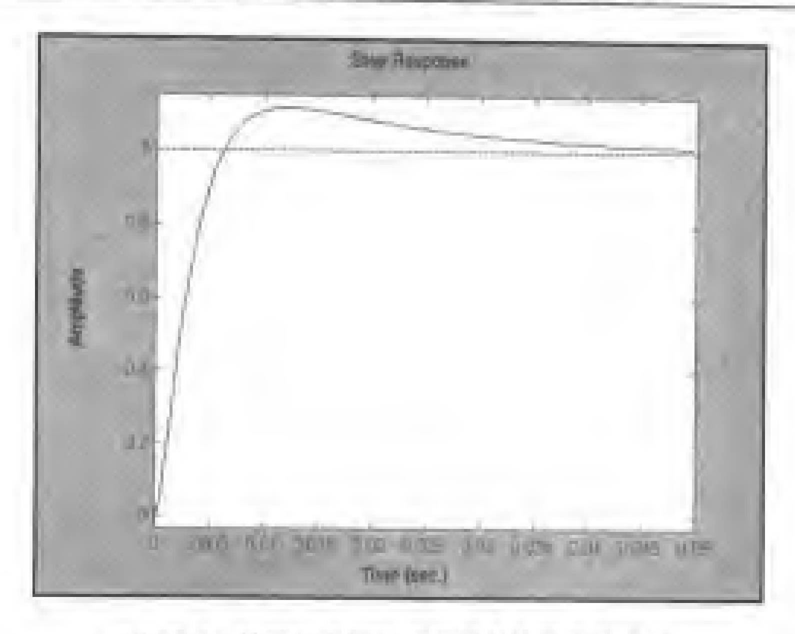


图 10-4 校正后电机定位系统的阶跃响应曲线

求解过程：本例的解题步骤分为以下几部分：

1．首先绘制原系统的 Bode 图和阶跃响应、频率响应（见图 10-5）

根据这些曲线分析与期望闭环系统的差异，大致确定校正装置的类型。在 MATLAB 环境下执行以下代码（可以在 MATLAB Command Window 下直接执行，但建议存为 M 文件，以便修改。以下同）：

如图 10-5 所示，左半部分是系统的 Bode 图，横坐标为频率，单位为 rad/s，纵坐标分别是增益 dB 值和相角°；右边上半部分是系统的阶跃响应曲线，下半部分是未取对数的闭环系统频率响应曲线。

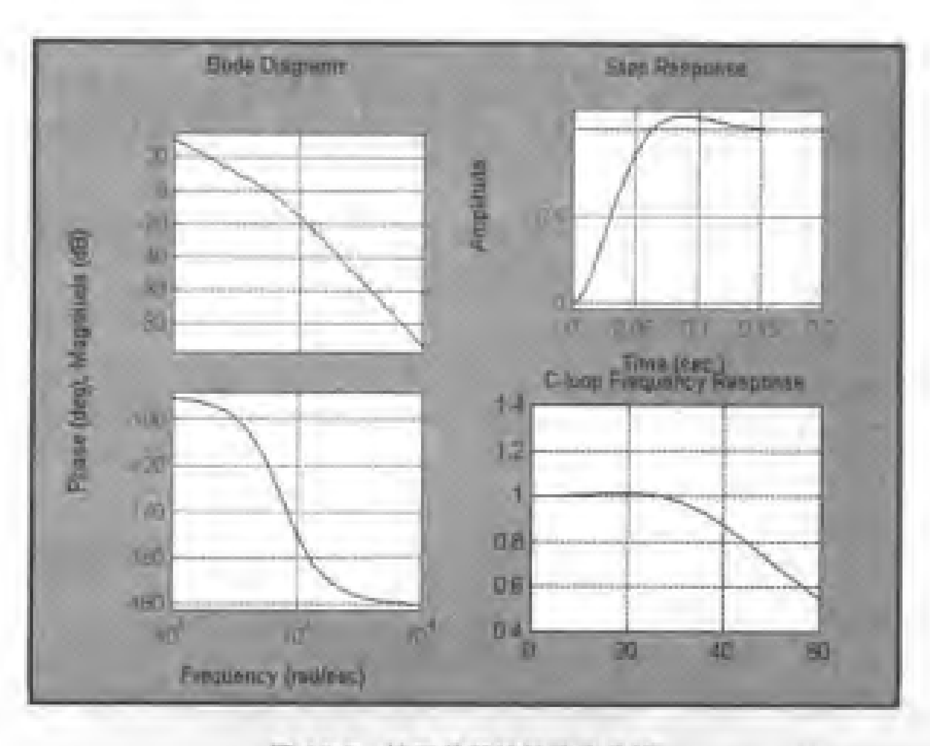


图 10-5 校正前系统的响应曲线

```
J=3.2284E-6;
b=3.5077E-6;
K=0.0274;
R=4;
L=2.75E-6;
num=[0 0 0 K];
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2) 0];
numc=num;
denc=den+num;
w=logspace(0,4,101);
wt=0:0.01:60;
t=0:0.001:0.2;
g=freqs(numc,denc,wt);
mag=abs(g);
subplot(121),
bode(num,den,w);
subplot(222),
step(numc,denc,t);
grid;
subplot(224),
plot(wt,mag);
title('C-loop Frequency Response');
grid;
```

从系统校正前的阶跃响应曲线可以看到，虽然超调量基本满足要求，但系统的响应速度太慢，过渡过程时间约为 150ms。不过我们首先要满足对扰动静态无差的要求。因为如果最后解决静态误差的话，开始添加的频率校正环节就有可能不再满足设计要求了。这也是频域系统设计中一定要注意的问题。

2．在系统的开环传递函数中添加一个积分器 1/s，满足系统对静态误差的要求

紧接着上一步的窗口或 M 文件，键入以下代码：

```
numi=1;
deni=[1 0];
numiol=conv(num,numi);
deniol=conv(den,deni);
bode(numiol,deniol,w)
```

得到初步校正后系统的 Bode 图如图 10-6 所示。其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。

3．求解期望开环频率响应的截止角频率和曲线特性

设计要求超调量小于 16%，过渡过程时间小于 40ms，根据 10.1 节提供的经验公式，在 MATLAB Command Window 下输入下列代码：



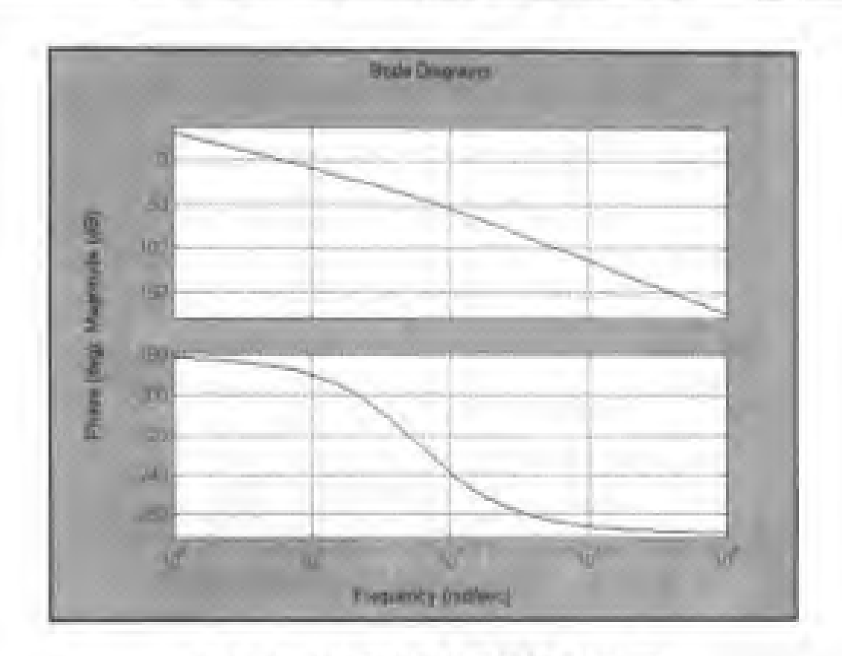


图 10-6　加入积分器后系统的 Bode 图

```
ts=0.04;
sigmac=16;
wc=9/ts;
gama=2000/(sigmac+20);
```

得到截止角频率和相角裕量为：

```
wc=225
gama=55.5556
```

考虑到原系统阶次较高，取 ω_c=250rad/s，γ=50°。对照图 10-6，发现 ω_c=250rad/s 左右时系统的幅频特性约为-60dB，相频特性约为-260°。并且，从幅频特性曲线的斜率上可以看出，系统大约在 60rad/s 左右有一个极点，这个低频段与中频段结合部位的极点对系统的稳定性影响较大，应设法消去。

4．设计校正装置消去低频段与中频段结合部位的极点

准备采用积分加导前的校正装置(s+60)/s，消去 60rad/s 左右的极点。紧接上一步，输入如下代码：

```
numpi=[1 60];
denpi=[1 0];
%闭环传递函数
numpiol=conv(numpi,num);
denpiol=conv(denpi,den);
bode(numpiol,denpiol,w)
```

校正后系统的 Bode 图如图 10-7 所示。

从图 10-7 中可以看到，系统的幅频特性曲线是斜率约为-2 的直线，而相频特性曲线在 180°左右变化。因此，对照 ω_c=250rad/s、γ=50°的要求，需要在 ω_c=250rad/s 左右补偿 50°的相角。

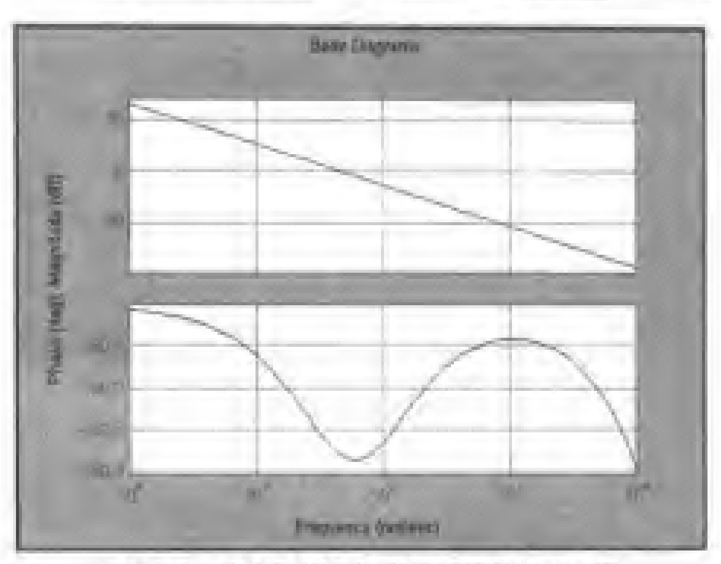


图 10-7　消去低频段极点后系统的 Bode 图

5. 满足 ω_c=250rad/s、γ=50° 的要求

根据 10.1 节的经验公式，求得校正装置的传递函数为

$$\frac{0.0189s+1}{5.8776e-4s+1}$$

紧接上一步，键入以下代码：

```
a=(1 - sin(gama*pi/180))/(1 + sin(gama*pi/180));
T=1/(wc*sqrt(a));
numpil = conv([1 60],[T 1]);
denpil = conv([1 0],[a*T 1]);
numpilol = conv(numpil,num);
denpilol = conv(denpil,den);
w = logspace(2,3,101);
bode(numpilol,denpilol,w)
```

校正后系统的 Bode 图如图 10-8 所示。

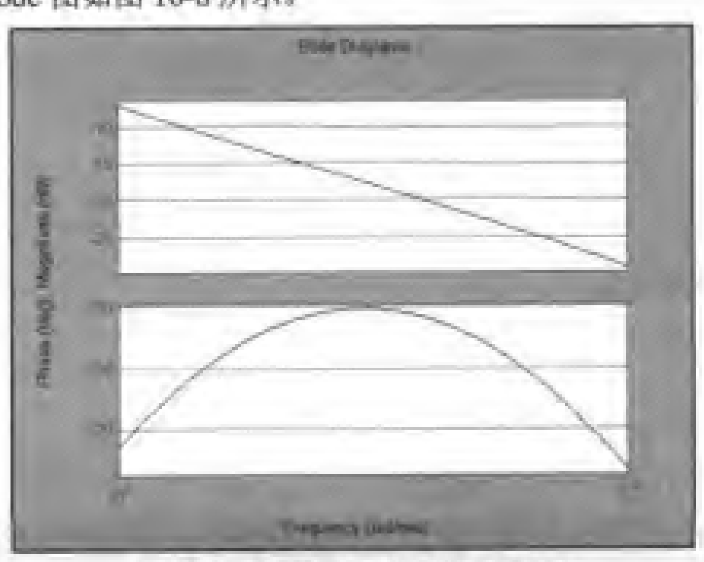


图 10-8　校正相角后的系统 Bode 图

从图 10-8 中可以看到，系统在 ω_c=250rad/s 时相角不低于-120°，满足了初步设计要求。但此时的幅频特性约为-20dB，与期望值相差-20dB。不过这一点改善起来比较容易，只需增大系统的低频增益即可。

6．改善系统的低频增益并初步检验校正结果

与-20dB 相对应的系统增益为 K=10，因此，将系统的开环传递函数乘以 10，并绘制其变化阶跃响应曲线以检验初步设计的结果。紧接上一步，键入以下代码：

```
kpid = 10;
bode(kpid*numpilol,denpilol,w)
[numpilcl,denpilcl] = cloop(kpid*numpilol,denpilol,-1);
t = 0:0.001:0.1;
step(numpilcl,denpilcl)
```

此时系统的 Bode 图和阶跃响应图如图 10-9 所示。

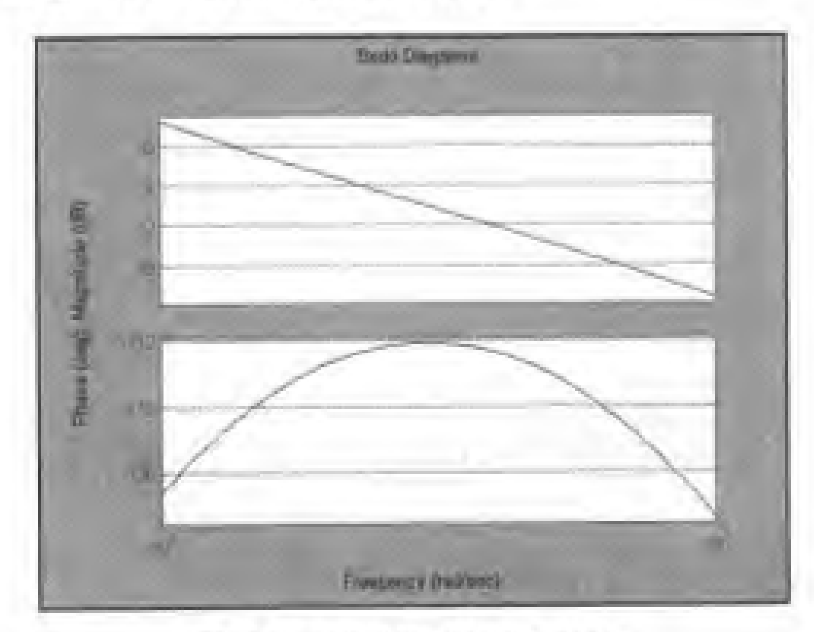


图 10-9　补偿增益后的系统 Bode 图

其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。

从图 10-10 细心的初步设计结果中可以看到，此时系统的过渡过程时间约为 30ms，好于预期的 40ms；但超调量太大，约为 28%，比预期的 16%多了近 3/4，这在工程上是无法接受的。因此，还需要进一步设计，目标是减小系统的超调量。

根据 10.1 节的经验公式，将系统的相角裕量扩大到 70°，并相应地在截止频率方面作一定补偿，令 ω_c=300rad/s。紧接上一步，键入以下代码：

```
%相角裕量
gama=70;
%截止角频率
wc=300;
%超前校正
a=(1 - sin(gama*pi/180))/(1 + sin(gama*pi/180));
```

```
T=1/(wc*sqrt(a));
numpil=conv([1 60],[T 1]);
denpil=conv([1 0],[a*T 1]);
%闭环传递函数
numpilol=conv(numpil,num);
denpilol=conv(denpil,den);
%绘制 Bode 图
w=logspace(2,3,101);
kpid = 5;
bode(kpid*numpilol,denpilol,w)
```

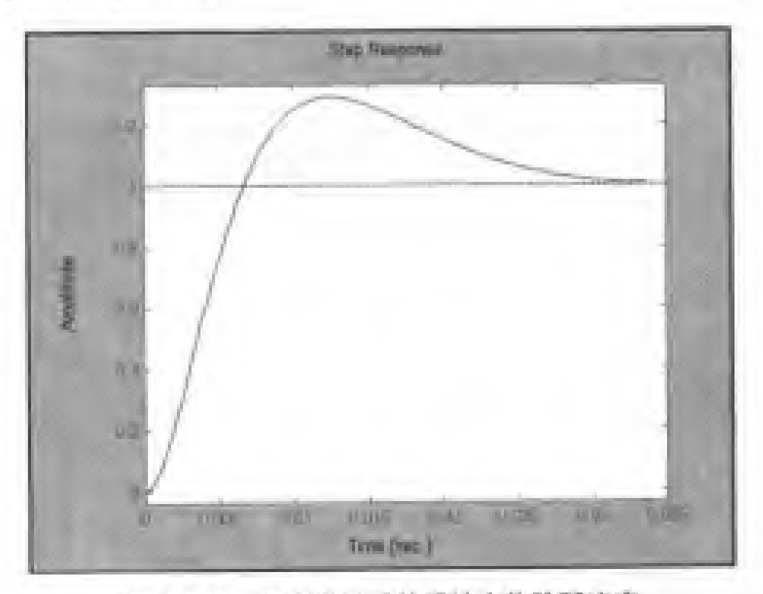


图 10-10 初步设计后的系统变化阶跃响应

此时系统的 Bode 图如图 10-11 所示。

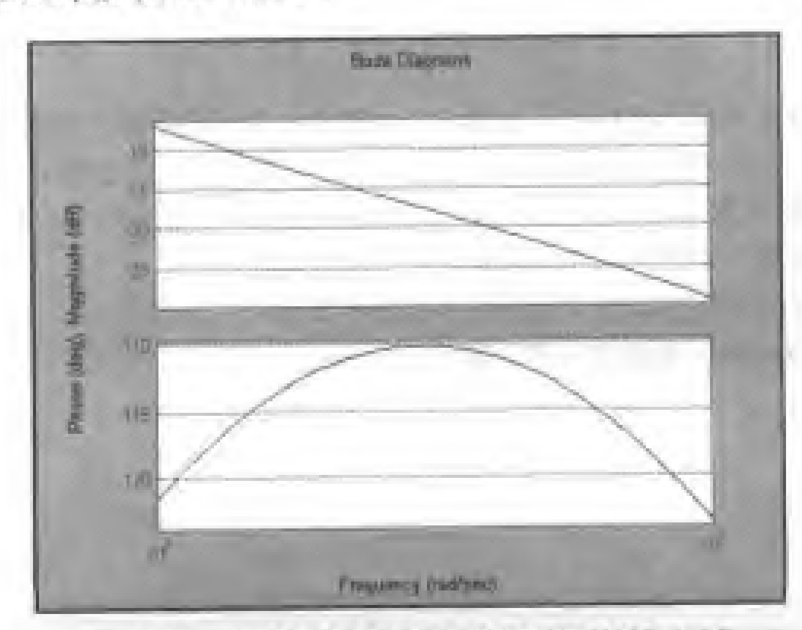


图 10-11 增大相角裕量和截止频率后的系统 Bode 图

从图 10-11 来看，与图 10-8 相同，都是相角裕量满足要求但低频增益过低，与期望值相差约-18dB。

7．补偿增益并最终检验校正结果

在 MATLAB Command Window 下计算求得：

```
10^0.9
ans = 7.9433
```

因此，选择增益 K＝8，输入以下代码：

```
kpid=8;
bode(kpid*numpilol,denpilol,w);
%闭环系统传递函数
[numpilcl,denpilcl]=cloop(kpid*numpilol,denpilol,-1);
t=0:0.001:0.1;
%绘制阶跃响应曲线
step(numpilcl,denpilcl)
```

系统最终的 Bode 图如图 10-12 所示。

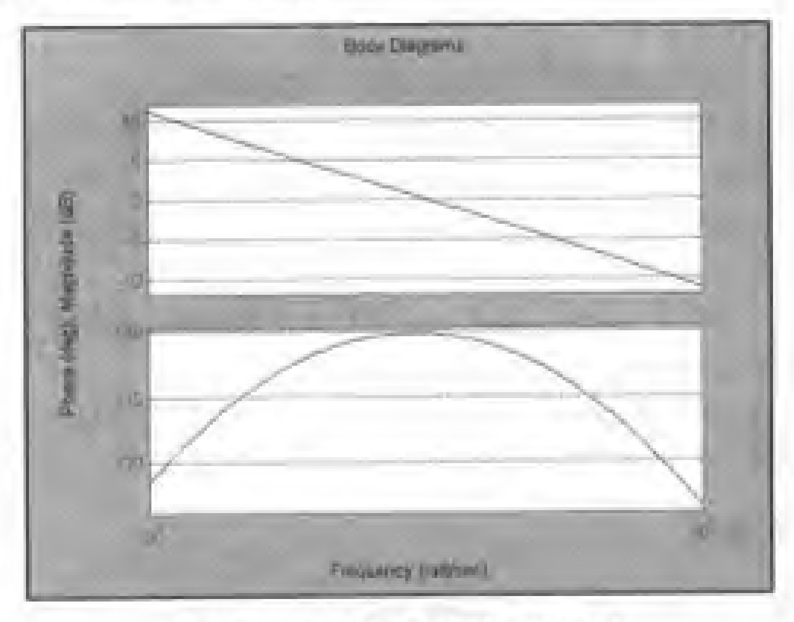


图 10-12 电机定位系统校正最终结果

系统最终的阶跃响应和校正装置曲线请参阅图 10-4 及结果，各项指标均满足系统最初的设计要求。

小结：从本例的解题过程可以看出，MATLAB 强大的绘图功能非常适合用来进行控制系统的频率域校正。在求解校正装置时可以逐步试探，所需要的性能指标从图中基本都可以读出，或者只需要根据经验公式进行简单的计算。而手工计算时不可能绘制出如此精确的图形，只能靠数值计算，这样做一方面计算量比较大，另一方面也不够直观，有可能出现较大的偏差。当然，使用 MATLAB 进行频率域控制系统设计也不会一帆风顺，本例的求解过程就大致分为初步设计和二次设计两部分。不过 MATLAB 的优点在于可以应用以前的结果迅速更改设计参数，不必重新进行繁琐的环境初始化和数据输入。因此 MATLAB 在控制领域被称为是“草

稿纸式的演算语言”，就是赞扬它这种继承性的开放式结构。本例还用到了一个新的 MATLAB 函数 cloop ( )，其功能是根据系统的开环传递函数和反馈类型求解系统的闭环传递函数。通过上边的代码读者基本就可以理解其调用格式和参数设置，详细情况请读者参阅 MATLAB 帮助。

10.3 串联校正

串联校正的主要内容是 PID 校正，即比例积分微分校正。事实上，在工程领域中应用简单串联校正的场合，80%都是 PID 校正。因此，本节就以 PID 校正为例，详细阐述串联控制的原理和应用。

在生产过程系统控制的发展历程中，PID 校正是历史最悠久、生命力最强的基本控制方式。在 20 世纪 40 年代以前，除在最简单的情况下可以采用开关控制外，它是惟一的控制方式。此后，随着科学技术的发展特别是数字式计算机的诞生和发展，涌现出许多新的控制方式。然而直到现在，PID 校正由于它自身的优点仍然是得到最广泛应用的基本控制方式。简单说来，PID 校正具有以下优点：

(1) 原理简单，使用方便。

(2) 适应性强，可以广泛的应用于各种工业过程控制领域。

(3) 鲁棒性强，即其控制品质对被控对象特性的变化不大敏感。这也是 PID 校正获得广泛应用的最主要的原因。一方面，它成本低廉，易于操作；另一方面，对于绝大部分控制对象，可以不必深究其模型机理，直接应用 PID 校正，其较强的鲁棒性保证了加入校正装置的系统的性能指标基本能满足要求。

当然，PID 校正也有其局限性。对于大延迟系统和性能指标要求特别高的系统 PID 校正就无能为力了，这就需要考虑更先进的控制方法。

本节将首先通过一个简单的例子介绍 PID 校正的原理及其在 MATLAB 下的实现方法，然后再根据一个串联校正实例，详细介绍 PID 校正的工程应用。

10.3.1 PID 校正概述

如前所述，PID 校正就是比例（Proportional）积分（Integral）微分（Derivative）校正的简称。传统的 PID 调节器的动作规律是

$$u = K_P e + K_I \int e\,\mathrm{d}t + K_D \frac{\mathrm{d}e}{\mathrm{d}t}$$

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

这是典型的按偏差控制的负反馈结构。其中 e 是偏差，即输出量与设定值之间的差；u 是控制量，作用于被控对象并引起输出量的变化。K_P 是比例增益系数，其控制效果是减小响应曲线的上升时间及静态误差，但无法做到消除静态误差，因此，单纯的 P 校正是有差调节，一般不会单独使用。K_I 是积分增益系数，其控制效果是消除静态误差，I 校正是无差调节；但它会延长过渡过程时间，增大超调量，甚至影响系统的稳定性，因此，一般也不会单独使用。

K_D 是积分增益系数，其控制效果是增强系统的稳定性，减小过渡过程时间，降低超调量。K_P、K_I、K_D 与系统时间域性能指标之间的关系见表 10-1。

表 10-1　PID 调节参数与系统时间域性能指标间关系

参数名称	上升时间	超调量	过渡过程时间	静态误差
K_P	减小	增大	微小变化	减小
K_I	减小	增大	增大	消除
K_D	微小变化	减小	减小	微小变化

上表的意义是 PID 参数增大时各系统性能指标的变化情况。当然，各参数与性能指标之间的关系不是绝对的，只是表示一定范围内的相对关系。因为各参数之间还有相互影响，一个参数变了，另外两个参数的控制效果也会改变。因此，在设计和整定 PID 参数时，上表只起一个定性的辅助作用。下面以表 10-1 为基础，希望通过这个例子，使读者了解 PID 校正的基本功能在 MATLAB 下实现的方法。由于空间有限，一些 PID 参数整定过程的图表就略去了，不过思路是相同的，读者可以自己掌握。

考虑图 10-13 我们熟悉的弹簧—阻尼系统，传递函数 $G(s)$如下，参数为 $M=1\text{kg}$ ，$b=10\ \text{N}\cdot\text{s/m}$ ，$k=20\ \text{N/m}$ ，$F(s)=1$ 。试设计不同的 P、PD、PI、PID 校正装置，使相同的响应曲线满足：

(1) 较快的上升时间和过渡过程时间。

(2) 较小的超调量。

(3) 静态误差为零。

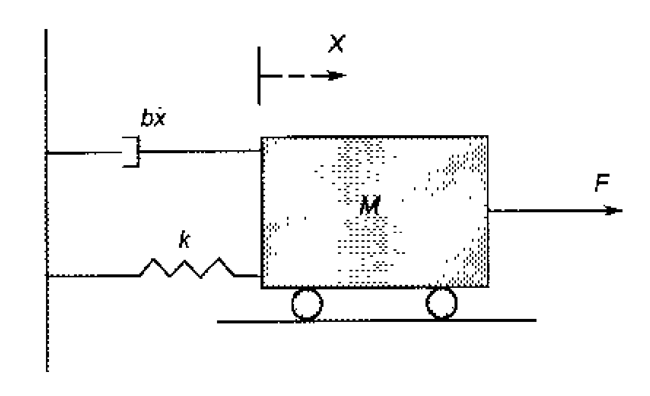


图 10-13　弹簧—阻尼相同示意图

结果：求得的校正装置为：

$$G(s)=\frac{X(s)}{F(s)}=\frac{1}{Ms^2+bs+k}$$

P 校正：K_P=300；

PD 校正：K_P=300，K_D=10；

PI 校正：K_P=30，K_I=70；

PID 校正：K_P=350，K_I=300，K_D=50。

求解过程：

本例的解题步骤分为以下几部分：

1．求解未加入校正装置的系统开环阶跃响应

根据系统的开环传递函数，在 MATLAB Editor/Debugger 下编辑下述代码：

```
clear;
num=1;
den=[1 10 20];
step(num,den)
```

得到系统的开环阶跃响应如图 10-14 所示。

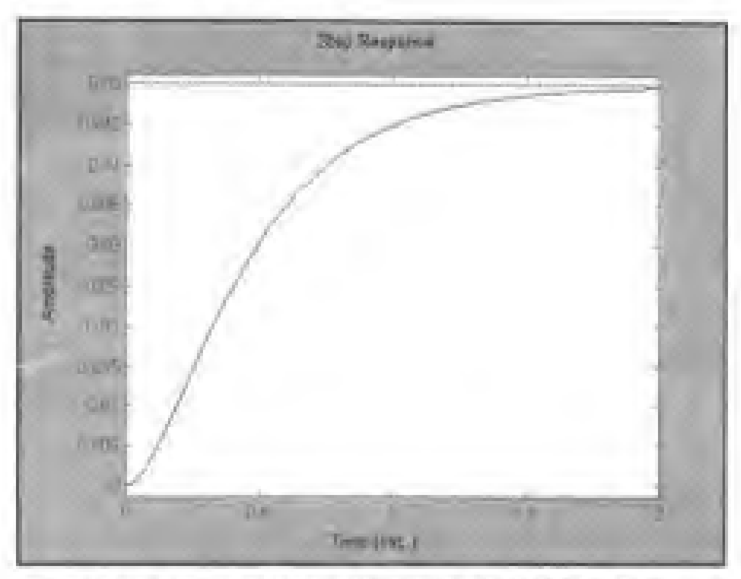


图 10-14　未加入校正装置时系统的开环阶跃响应

从图 10-14 中可以看出，系统的开环响应曲线未产生振荡，属于过阻尼性质。这类曲线一般响应速度都比较慢。果然，从图中读出，系统的上升时间约为 1s 左右，过渡过程时间约为 1.5s 左右。对于一个虎克常数为 20N/m 的弹簧系统来说，这样的响应速度确实太慢了，很难满足迅速定位的需要。并且，另外一个关键性的问题是，系统在幅值为 1 的阶跃响应输入下稳态值仅为 0.05，静态误差达 0.95，远不能满足跟随设定值的要求。这是因为系统传递函数分母的常数项为 20，也就是说系统对直流分量的增益是 1/20。因此，时间趋于无穷，角频率趋于零时，系统的稳态值就趋于 1/20=0.05。为了大幅度降低系统的静态误差，我们首先想到 P 校正。

2．P 校正装置设计。

从表 10-1 中我们看到增大 K_P 可以降低静态误差、减小上升时间和过渡过程时间，因此首先选择 P 校正，也就是在系统中加入一个比例放大器。此时系统的闭环传递函数为：

$$G_c(s)=\frac{K_p}{s^2+10s+(20+K_p)}$$

此时系统的静态误差为 K_P/（20+K_P）。本着尽量减小静态误差的原则，我们取系统的比例增益 K_P=300，这样可以把系统的静态误差降低到 0.06 左右。但并不是说系统的比例增益 K_P 可以无限制的越大越好，而是要受到实际系统的物理条件和放大器的实际情况限制，一般也就是取到几十或几百的量级。另外，增大系统的比例增益还可以改善系统的快速性。紧接上一步，输入以下代码：

```
Kp=300;
num=[Kp];
%特征方程
den=[1 10 20+Kp];
t=0:0.01:2;
%系统阶跃响应
step(num,den,t)
```

得到加入 P 校正后系统的闭环阶跃响应如图 10-15 所示。

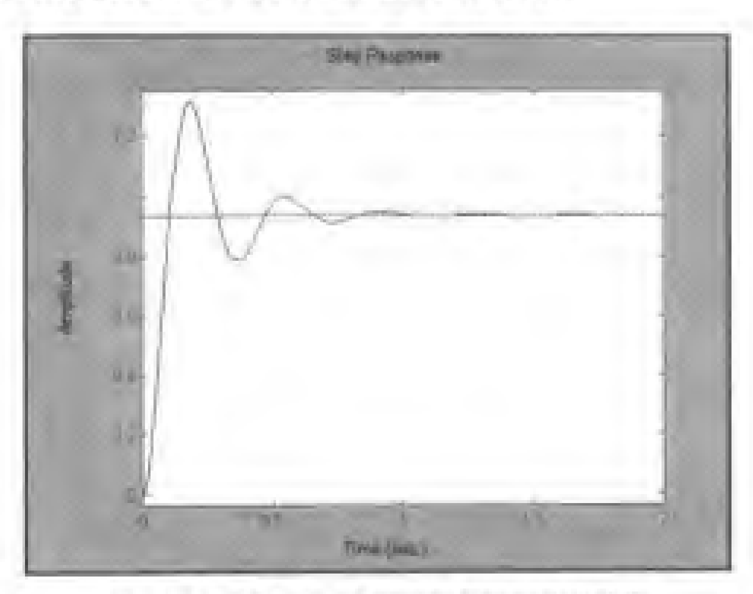


图 10-15 P 校正后系统的闭环阶跃响应曲线

从图 10-15 可以看到，系统的稳态值在 0.94 左右，静态误差约为 0.06，这和最初的设计要求是一致的。并且，曲线的形状从过阻尼型变为衰减振荡型，衰减比（响应曲线第一个峰值和第二个峰值之比）大于 4∶1。因此，系统的快速性应该有一定的改善。从图中读出，系统的上升时间不超过 0.2s，过渡过程时间不超过 0.7s，验证了表 10-1 有关参数变化对系统性能影响的说法。

不过曲线由过阻尼改为衰减振荡又产生了另外一个问题，就是系统的超调量达到 30%以上，第一个峰值振荡幅度过大。应该寻找另外的校正方式解决这个问题，下面我们尝试使用 PD 校正。

3. PD 校正装置设计

从表 10-1 中我们看到增大 K_D 可以降低超调量、减小过渡过程时间，对上升时间和静态误差影响不大。因此可以选择 PD 校正，也就是在系统中加入一个比例放大器和一个微分器。此时系统的闭环传递函数为：

$$G_c(s)=\frac{K_D s+K_P}{s^2+(10+K_D)s+(20+K_P)}$$

仍选择 K_P=300；K_D 一般选择为系统振荡周期倒数的 8 倍左右，也可以根据系统性能指标用 MATLAB 不断绘图校正。这里略去绘图选择的过程，给出 K_D=10。代码如下：

```
Kp=300;
```

```
Kd=10;
%校正装置
num=[Kd Kp];
%特征方程
den=[1 10+Kd 20+Kp];
t=0:0.01:2;
step(num,den,t)
```

得到加入 PD 校正后系统的闭环响应如图 10-16 所示。

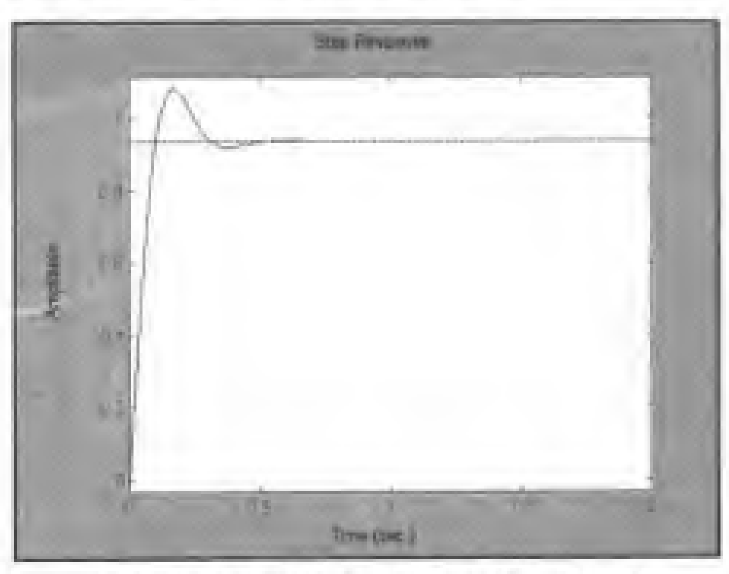


图 10-16 PD 校正后系统的闭环阶跃响应曲线

从图 10-16 可以看出，加入微分器后系统的响应曲线形状仍是衰减振荡型，但振荡次数显著减少，并且超调量也降低了不少。这就进一步验证了表 10-1 中有关增大 K_D 可以增强系统稳定性的说法。从系统的上升时间和静态误差来看，K_D 的变化对其影响不大。现在的问题就是静态误差不为零，这需要 P_I 校正来解决。

4．P_I 校正装置设计

从表 10-1 中我们看到增大 K_I 可以消除静态误差，因此可以选择 P_I 校正，也就是在系统中加入一个比例放大器和一个积分器。此时系统的闭环传递函数为：

$$G_c(s)=\frac{K_P s+K_I}{s^3+10s^2+(20+K_P)s+K_I}$$

考虑到 K_I 的作用，可以大幅度降低 K_P，取 K_P=30。K_I 可以选得大一些，逐步降低。这里取 K_I=70。输入代码：

```
t=0:0.01:2;
step(num,den,t)
Kp=30;
Ki=70;
num=[Kp Ki];
den=[1 10 20+Kp Ki];
```

```
%绘制阶跃响应曲线
t=0:0.01:2;
step(num,den,t)
```

得到加入 PI 校正装置后系统的响应曲线如图 10-17 所示。

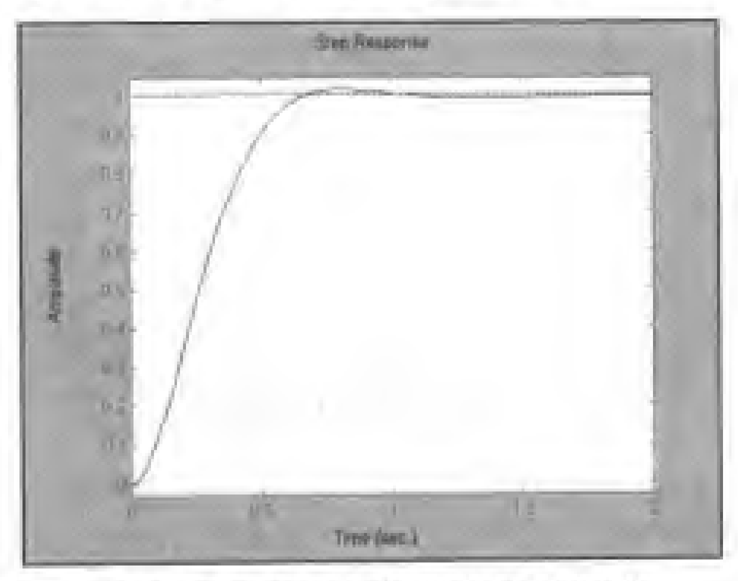


图 10-17　PI 校正后系统的闭环阶跃响应曲线

从图 10-17 中可以看到，加入 PI 校正后系统的稳态值为 1，即静态误差为零，系统的输出量最终能够无差的跟踪设定值的变化。不过由此又产生了另一个问题，系统的过渡过程时间显著增加，快速性降低。这也是使用积分器带来的副作用。在这种情况下，如果希望系统各方面的性能指标都达到一个满意的程度，一般都要采用典型的 PID 校正。

5．PID 校正装置设计

对于本例这种机械控制系统，采用 PID 校正一般都能取得满意的控制结果。其校正装置为：

$$G_c(s)=\frac{K_D s^2+K_P s+K_I}{s^3+(10+K_D)s^2+(20+K_P)s+K_I}$$

K_P、K_I 和 K_D 的选择一般是先根据经验公式确定一个大致的范围，然后通过 MATLAB 绘制的图形逐步校正。这里由于不考虑校正装置的可实现性，参数可以取的大一些。取 K_P=350，K_I=300，K_D=50。输入代码：

```
Kp=350;
Ki=300;
Kd=50;
%PID 校正装置
num=[Kd Kp Ki];
den=[1 10+Kd 20+Kp Ki];
%绘制阶跃响应曲线
t=0:0.01:2;
step(num,den,t)
```

得到加入 PID 校正装置后系统的响应曲线如图 10-18 所示。

图 10-18　PID 校正后系统的闭环阶跃响应曲线

从图 10-18 可以看到，系统的响应曲线几乎已经和阶跃函数本身差不多了。当然，这只是一种理想状态的控制情况。并且还由于本例是机械系统，不存在延迟，而且快速性较好的缘故，实际上是很难达到的。

不过我们也应该看到，对于一般的控制系统来说，应用 PID 控制还是比较有效的，而且基本不用分析被控对象的机理，只根据 K_P、K_I 和 K_D 的参数特性以及 MATLAB 绘制的阶跃响应曲线进行设计即可。

小结：一般说来，在工程控制领域常用的是 PD 校正、PI 校正和 PID 校正。如果对控制品质要求不高，有时也会用到 P 校正。关于参数整定，人们总结了许多经验图表和公式，比较著名的有动态特性参数法、稳定边界法、衰减曲线法以及 Ziegler-Nichols 经验公式等等。

但是在 MATLAB 环境下，我们甚至可以不借助这些经验公式，直接根据仿真曲线来选择 PID 参数。根据系统的性能指标和一些基本的整定参数的经验，选择不同的 PID 参数进行仿真，最终确定满意的参数。这样做一方面比较直观，另一方面计算量也比较小，并且便于调整。当然，在大多数情况下，根据经验公式进行计算还是有必要的，可以为我们指明参数选择的方向。

10.3.2　串联校正举例

上一小节讲述的 PID 校正主要是应用在时间域，相应的频率域，提法分别叫做超前校正（PD 校正）、滞后校正（PI 校正）和滞后超前校正（PID 校正）。

超前校正的校正装置传递函数为

$$G(s)=\frac{aTs+1}{Ts+1},\ a>1$$

其功能是在指定的频率附近产生相位超前角，提高系统的相角裕量。如果希望校正后的系统截止角频率为 ω，并在 ω 附近相对于原系统提供 θ 度的超前角，则

$$\omega=\frac{1}{\sqrt{a}T}\qquad a=\frac{1+\sin\theta}{1-\sin\theta}$$

滞后校正的校正装置传递函数为：

$$G(s)=\frac{Ts+1}{\beta Ts+1},\ \ \beta>1$$

其功能是只降低中频段和高频段的开环增益而不影响低频段。如果希望校正后的系统截止角频率为 ω，并在 ω附近原系统的增益为 L，则：

$$\frac{1}{T}=(\frac{1}{4}\sim\frac{1}{10})\omega \qquad L=20\lg\beta$$

滞后超前校正的校正装置传递函数为：

$$G(s)=\frac{T_2s+1}{\beta T_2s+1}\frac{aT_1s+1}{T_1s+1},\ \ a,\beta>1$$

其功能是增加相位稳定裕量，改善系统动态性能的同时降低中频段增益，改善系统的静态性能。一般在单独使用超前校正或滞后校正无法实现控制目标的情况下采用滞后超前校正。请看下面这个例子。

某控制系统固有部分传递函数 G（s）如下，试分别设计串联校正装置 K（s），满足下列要求：

(1) 要求开环本例系数 $K\geqslant100$，相角裕量 $\gamma\geqslant30°$ ，截止角频率 $\omega\geqslant45$rad/s;

(2) 要求开环本例系数 $K\geqslant100$，相角裕量 $\gamma\geqslant40°$ ，截止角频率 $\omega=5$rad/s;

(3) 要求开环本例系数 $K\geqslant100$，相角裕量 $\gamma\geqslant40°$ ，截止角频率 $\omega\geqslant20$rad/s;

$$G(s)=\frac{1}{s(0.1s+1)(0.01s+1)}$$

结果：求得的校正装置分别为：

(1) 超前校正：$K(s)=100\dfrac{0.063s+1}{0.0063s+1}$

(2) 滞后校正：$K(s)=100\dfrac{0.8s+1}{14.22s+1}$

(3) 滞后超前校正：$K(s)=100\dfrac{0.25s+1}{1.253s+1}\dfrac{0.137s+1}{0.0182s+1}$

分析：因为给出的要求都是频率域的指标，所以用超前或滞后的串联校正比较合适。当然，用 10.2 节的开环频率设计法也可以。相角裕量是保证系统稳定性的，设计过程中一般要留出 5°～12° 的滞后量；截止角频率是控制系统通频带的指标，设计完之后一般要用 10.1 节的经验公式校验一下系统的阶跃响应。

求解过程：

本例的解题步骤分为以下几部分：

1．求解原系统的响应曲线，对照性能指标作出分析

在 MATLAB Editor/Debugger 下编辑以下代码：

```
%清除内存变量
clear;
num=1;
```

```
den1=[1 0];
den2=[0.1,1];
den3=[0.01,1];
den=conv(den1,conv(den2,den3));
%系统闭环传递函数
[numc,denc]=cloop(num,den,-1);
t=0:0.01:1.5;
wt=0:0.5:60;
%系统频率响应数据
g=freqs(numc,denc,wt);
mag=abs(g);
%绘制图形
subplot(121),
bode(num,den);
subplot(222),
step(numc,denc,t);
subplot(224),
plot(wt,mag);
title('Frequency Response-Amplitude');
xlabel('Frequency-rad');
ylabel('Amplitude');
grid;
```

得到校正前系统的响应曲线如图 10-19 所示。

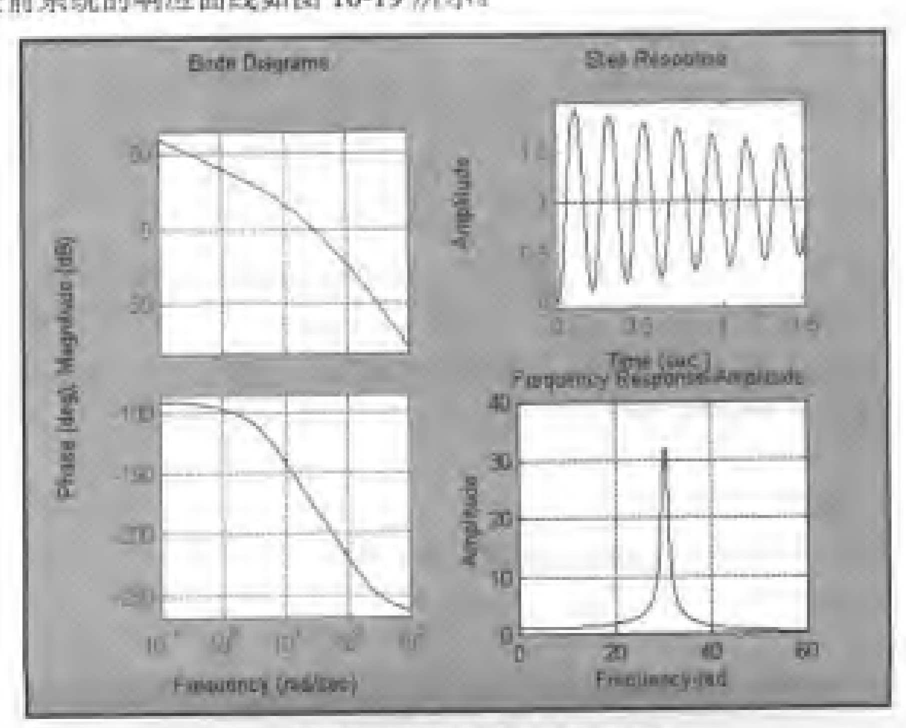


图 10-19 校正前系统的响应曲线



其中左半部分是系统的 Bode 图，横坐标为频率，单位 rad/s，纵坐标分别是增益 dB 值和相角(°)；右边上半部分是系统的阶跃响应曲线，下半部分是未取对数的闭环系统频率响应曲线。

从图 10-19 中可以看到，最明显的是系统的变化阶跃响应曲线振荡幅度非常剧烈，并且衰减频率很慢，系统濒临不稳定。对应系统的频率响应曲线上在 $\omega=30$rad/s 处有一个非常高的谐振峰，约为 32 左右。谐振峰高说明系统对该频率的输入容易引起共振，形成衰减很小的振荡曲线。而阶跃函数是包含一切频率分量的，因此对照阶跃响应曲线可以看出系统的振荡频率约为 $2\pi/32=0.2$s 左右。这样的系统是无法投入使用的，因此必须进行串联校正。

2．满足目标 1——超前校正

根据校正的目标，对照系统的 Bode 图可以发现：对于目标 1，$\omega=45$rad/s 时系统的相角已经接近 $-190°$，并且增益的 dB 值为负。因此应该在 $\omega=45$rad/s 处补充 50° 的相角超前量，使其此处增益的 dB 值接近于零。所以应采用超前校正。当然，考虑到对系统开环增益的要求，还应该将系统的增益值 K 乘以 100。紧接上一步，输入以下代码：

```
%相角裕量
gama=55;
%截止角频率
wc=50;
%超前校正装置
a=(1+sin(gama*pi/180))/(1-sin(gama*pi/180))
T=1/(wc*sqrt(a))
num=[0 0 0 100];
numa=[a*T 1]
dena=[T 1]
numao=conv(num,numa);
denao=conv(den,dena);
bode(numao,denao);
```

得到系统经超前校正后的 Bode 图如图 10-20 所示。

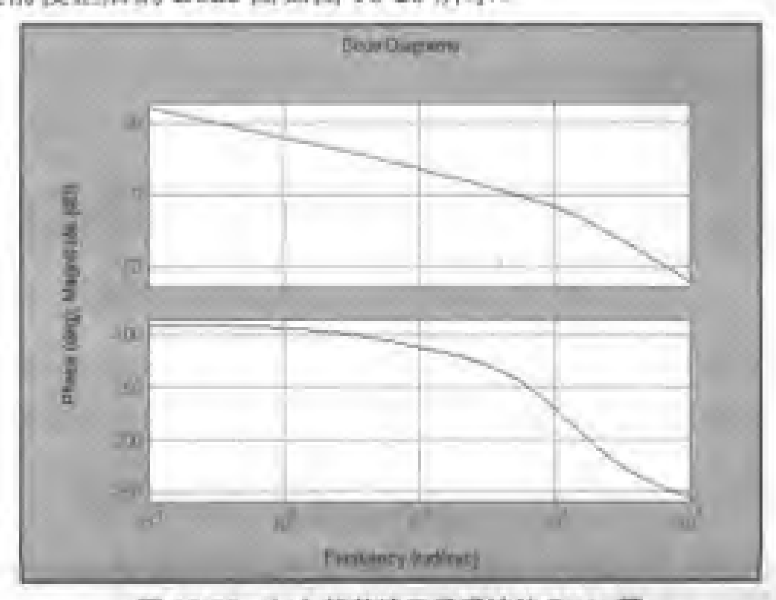


图 10-20　加入超前校正后系统的 Bode 图

其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。

对照图 10-20 和图 10-19，可以看到，系统的截止角频率从 $\omega=20$rad/s 后移到了约为 $\omega=45$rad/s 处，对应的相角约为-140°，相角裕量 40°，满足相角裕量 $\gamma\geq30°$ 的要求。这些结果验证了超前校正可以在一定的频率区域内提供相角超前量的说法。从 MATLAB Command Window 下可以读到校正装置的参数：

```
a = 10.0590
T = 0.0063
%校正装置传递函数多项式
numa = 0.0634     1.0000
dena = 0.0063     1.0000
```

numa 和 dena 就是求得的超前校正装置传递函数的分子和分母多项式系数。从上边的代码中还可以看到，这里选补偿的相角超前量 gama＝50°，暂定的截止角频率 ω=50rad/s。这是因为从图 10-19 的系统 Bode 图来看，$\omega=45$rad/s 处的相角约为-190°，要满足相角裕量 $\gamma\geq30°$ 的要求必须提供大于 40° 的相角超前量。考虑到传递函数间的相互影响取相角超前量为 50°。同理，取 $\omega=50$rad/s。同时还可以输入以下代码作时间域的验证：

```
[numac,denac]=cloop(numao,denao,-1);
step(numac,denac);
```

得到系统的阶跃响应曲线如图 10-21 所示。

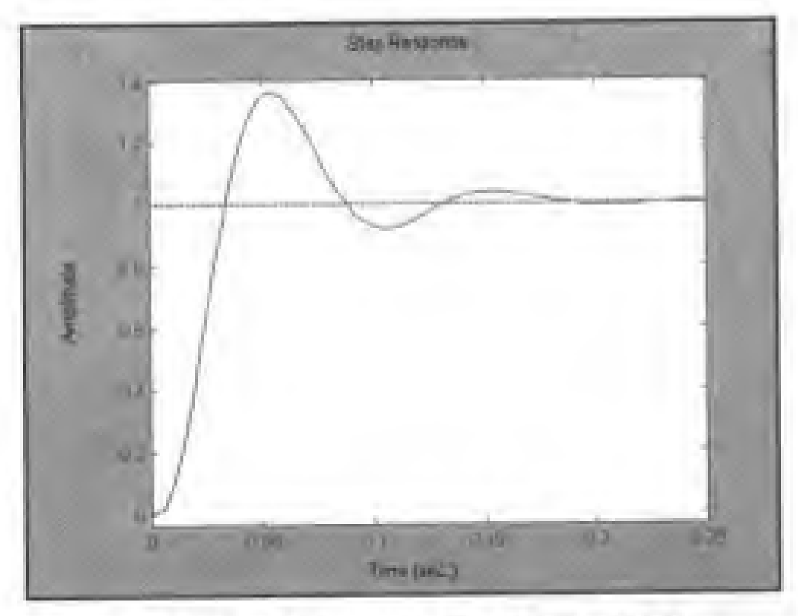


图 10-21　加入超前校正后系统的阶跃响应曲线

根据 10.1 节的经验公式，过渡过程时间 t_s=（4~9）/ ω。代入 $\omega=45$rad/s 后求得系统的过渡过程时间 t_s 在 0.09s 至 0.2s 之间。对照图 10-21，可以读出系统的过渡过程时间约为 0.18s 左右，满足设计要求。不过从图中还可以看到，系统的超调量仍然比较大，大约在 35%左右。前边说过，频率域的超前校正相当于时间域的 PD 校正。而从时间域性能指标上来说，PD 校正对完全抑制系统超调量的效果并不明显。不过，相对与未加校正时系统剧烈振荡的情况，系统的超调量还是降低了不少。

3. 满足目标 2——滞后校正

根据校正的目标，对照系统的 Bode 图可以发现：对于目标 2，ω=5rad/s 时系统的增益值约为 25dB，相角约为-120°，超过了相角裕量要求，可以通过适当引入滞后相角。但中频段系统增益过高，对于这种系统，超前校正是无能为力的，应该采用降低中频段增益的滞后校正。紧接上一步，输入以下代码：

```
wc=5;
g=25;
%滞后校正装置
beta=10^(g/20);
T=4/wc;
wt=logspace(-2,2);
numb=[T 1];
denb=[beta*T 1];
numbo=conv(num,numb);
denbo=conv(den,denb);
bode(numbo,denbo,wt);
```

得到系统经滞后校正后的 Bode 图如图 10-22 所示。

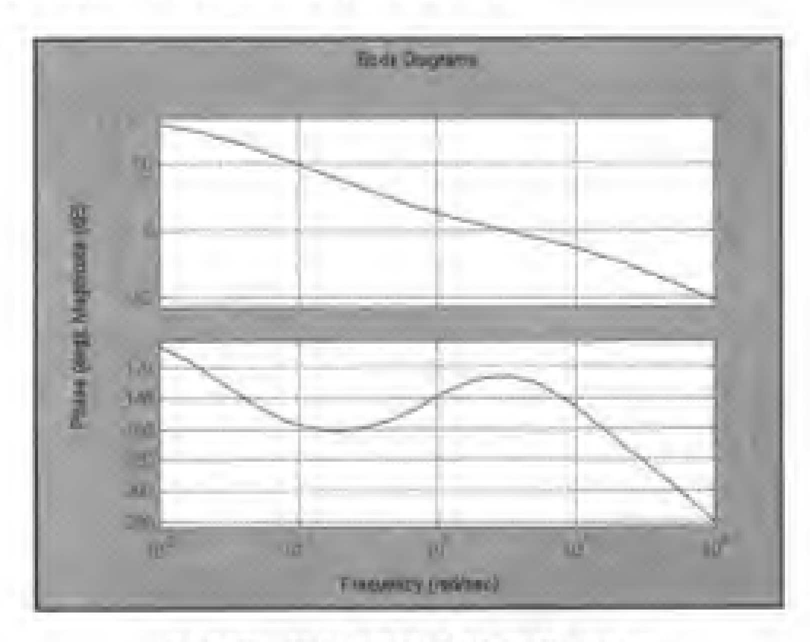


图 10-22 加入滞后校正后系统的 Bode 图

对照图 10-22 和图 10-19，可以看到，系统的截止角频率从 ω=20rad/s 前移到了约为 ω=5rad/s 处，对应的相角约为-130°，相角裕量 50°，满足相角裕量 $\gamma\geq 40°$ 的要求。这些结果验证了滞后校正可以降低中频段和高频段系统增益的说法。当然，校正装置在低频段引入了一定的相角滞后量，不过因为系统的截止频率在中频段，在满足相角裕量要求的情况下增加一些相角滞后量对系统的稳定性并不会产生太大影响。从 MATLAB Command Window 下可以读到校正装置的参数：

```
beta = 17.7793
```

```
T = 0.8
%校正装置传递函数多项式
numb = 0.8      1.0000
denb = 14.2234      1.0000
```

numb 和 denb 就是求得的滞后校正装置传递函数的分子和分母多项式系数。从上边的代码中还可以看到，这里暂定的截止角频率 ω=5，$T=4/\omega$；从图 10-19 中读出 ω=5rad/s 处系统的增益为 25dB。同时输入以下代码，可以求得加入滞后校正装置后系统的阶跃响应曲线：

```
[numbc,denbc]=cloop(numbo,denbo,-1);
step(numbc,denbc);
```

此时系统的阶跃响应曲线如图 10-23 所示。

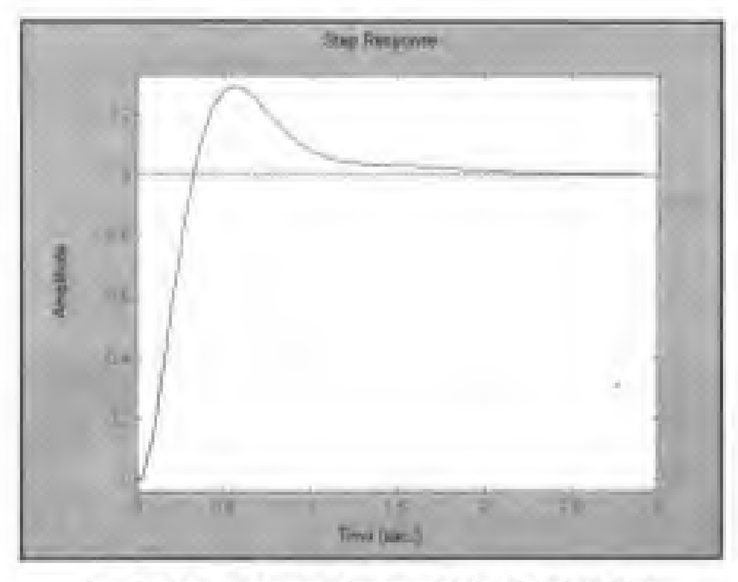


图 10-23　加入滞后校正后系统的阶跃响应曲线

根据 10.1 节的经验公式，代入 ω=5rad/s 后求得系统的过渡过程时间 t_s 为 0.8s～2s。对照图 10-21，可以读出系统的过渡过程时间约为 1.8s 左右，满足设计要求。如前所述，频率域的滞后校正相当于时间域 PI 校正，其效果是消除系统静态误差，降低中频段增益。

4．满足目标 3——滞后超前校正

根据校正的目标，对照系统的 Bode 图可以发现：对于目标 3，ω=20rad/s 时系统的增益值约为 0dB，相角约为-180°，相角裕量几乎为零。要达到校正目标，一方面要在 ω=20rad/s 处提供 40° 的相角裕量，另一方面又必须保证此时的系统增益仍然为零。这样的要求，单独的超前校正或滞后校正都是无法满足的。因此，必须采用综合这两种校正优点的滞后超前校正。首先设计超前校正装置，紧接上一步，输入下列代码：

```
%相角裕量
gama=50;
%截止角频率
wc=20;
%校正装置
a=(1+sin(gama*pi/180))/(1-sin(gama*pi/180))
```




```
T1=1/(wc*sqrt(a))
%系统传递函数
numa=[a*T1 1]
dena=[T1 1]
numao=conv(numa,num);
denao=conv(dena,den);
bode(numao,denao,wt);
```

得到系统的 Bode 图如图 10-24 所示。

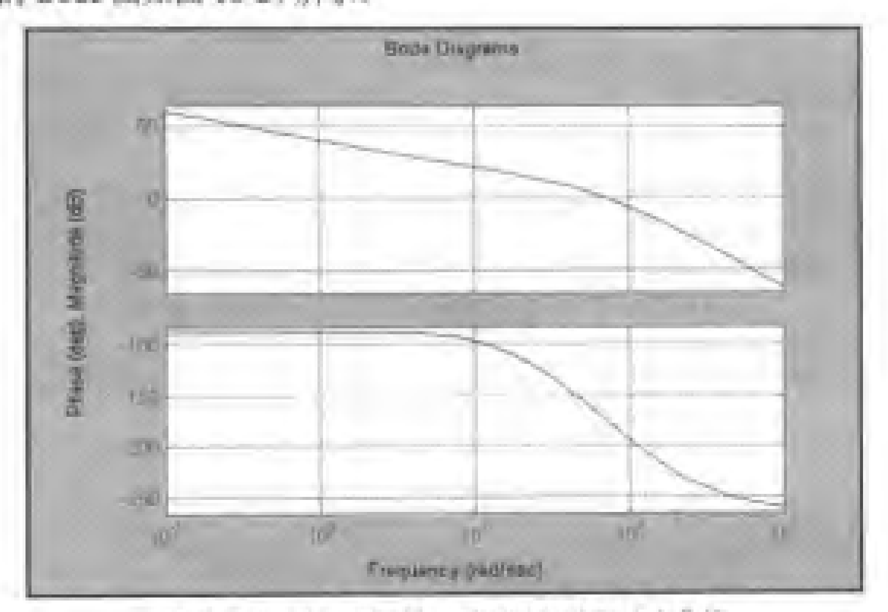


图 10-24　加入超前校正后系统的阶跃响应曲线

对照图 10-24 和图 10-19，可以看到，系统的截止角频率从 $\omega=20$rad/s 后移到了约为 $\omega=80$rad/s 处。$\omega=20$rad/s 处系统的增益值约为 15dB，系统的相角约为-130°。如果能把系统的响应曲线在 $\omega=20$rad/s 处下移 15dB 同时不影响低频段增益值，就能同时满足截止角频率 $\omega=20$rad/s、相角裕量 $\gamma\geqslant40°$ 和开环增益 $K\geqslant100$ 的要求。这正是滞后校正的功能，因此，继续设计一个滞后校正装置，将系统在 $\omega=20$rad/s 处的增益下移 15dB。紧接上一步，输入以下代码：

```
g=14;
%滞后校正装置
beta=10^(g/20);
T=5/wc;
wt=logspace(-1,3);
%系统传递函数
numb=[T 1];
denb=[beta*T 1];
numo=conv(conv(num,numa),numb);
deno=conv(conv(den,dena),denb);
bode(numo,deno,wt);
```

得到加入滞后校正后系统的响应曲线如图 10-25 所示。

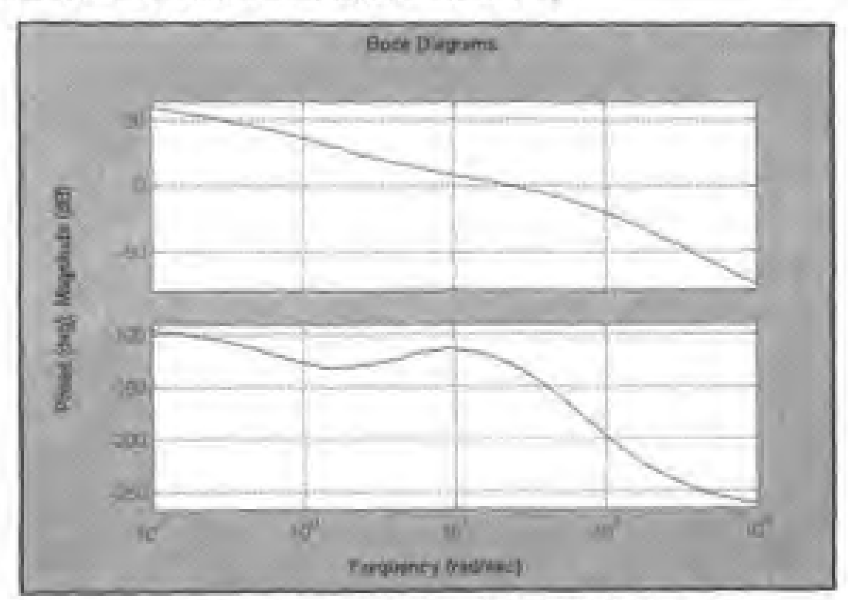


图 10-25 加入滞后超前校正装置后系统的 Bode 图

从 MATLAB Command Window 下可以读到超前校正和滞后校正的参数：

```
a = 7.5495
T = 0.0182
numa = 0.1374      1.0000
dena = 0.0182      1.0000
beta = 5.0000
T = 0.25
numb = 0.2500      1.0000
denb = 1.2500      1.0000
```

其中 numa 和 dena 是求得的超前校正装置传递函数的分子和分母多项式系数，numb 和 denb 是求得的滞后校正装置传递函数的分子和分母多项式系数。关于代码中各参数选择的理由和前面单独设计超前校正和滞后校正的理由基本一致，这里就不再赘述了。当然，有些是多次绘制响应曲线后选择的最佳值。从最后的效果来看，图 10-25 中系统的低频段增益超过 50dB，截止角频率 $\omega=20$rad/s，对应相角-130°，满足相角裕量 $\gamma\geqslant40°$ 和开环增益 $K\geqslant100$ 的要求。还可以得到此时系统的阶跃响应曲线如图 10-26 所示。

```
[numc,denc]=cloop(numo,deno,-1);
step(numc,denc);
```

此时系统的时间域动态性能就比较令人满意了。一方面是系统的超调量约为 20%，并且振荡次数不超过 2 次；另一方面系统的上升时间约为 0.1s，过渡过程时间不超过 0.4s，快速性也有一定保证。

前边曾经提到，频率域的滞后超前校正相当于时间域的 PID 校正。一般来说，普通的动态或静态性能指标要求通过调节 PID 校正的参数都可以实现。因此，如果发现单纯的超前校正或滞后校正不能满足设计要求的话，可以尝试应用滞后超前校正，一般都能得到较为满意的结果。

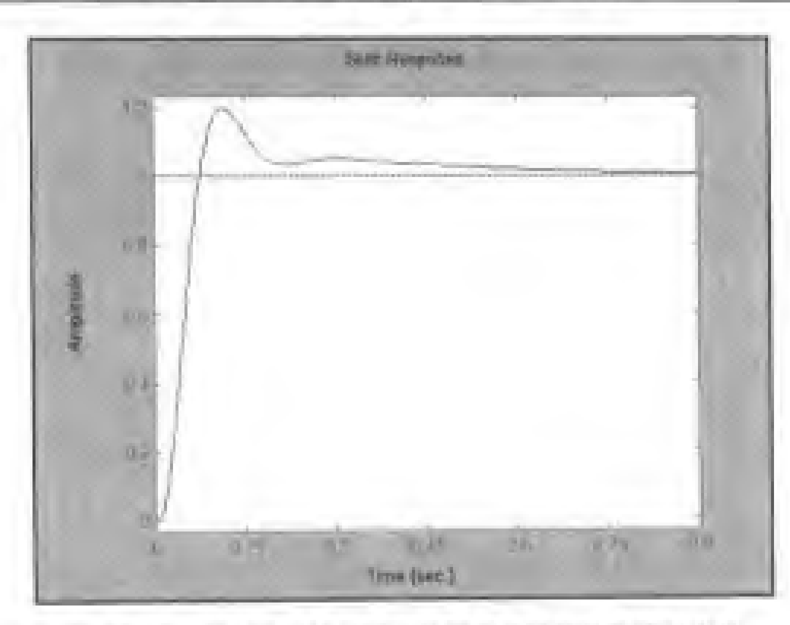


图 10-26 加入滞后超前校正装置后系统的阶跃响应曲线

小结：仔细比较本例不同的性能指标要求和校正装置实现的方式，我们可以发现下面这个规律：如果希望的截止角频率明显高于原系统的截止角频率（如 45rad/s 和 20rad/s），一般采用超前校正，在新的截止角频率处提供一定的相角超前量；如果希望的截止角频率明显低于原系统的截止角频率（如 5rad/s 和 20rad/s），并且新的截止角频率处系统的增益值大于 0dB，则一般采用滞后校正，将新的截止角频率处增益下调为零；如果希望的截止角频率和原系统的截止角频率差不多，但相角裕量要求较高，一般采用滞后超前校正，在新的截止角频率处提供一定的相角超前量，并将其增益下调为零。

通过上面这个例子，我们可以发现使用 MATLAB 来实现这样的工作是非常方便的。并且，大量的图形使得设计者可以随时掌握系统的情况，修正其设计方案，这也算是一种解决问题的负反馈吧。

10.4 根轨迹校正

事实上，除了用来判断系统稳定性以及参数变化对系统性能的影响之外，根轨迹还可以作为校正控制系统的手段使用。

10.4.1 Rltool 环境概述

MATLAB 提供了一个辅助设计闭环系统根轨迹的仿真软件 Rltool，可以用来进行根轨迹校正。与著名的 Simulink 一样，Rltool 也采用了可视化的设计方法，在 MATLAB Command Window 下键入 rltool，即可进入 Rltool 的仿真界面如图 10-27 所示。

在 Rltool 环境下只需设计很少的环境参数即可进行根轨迹的仿真和校正。并且，根轨迹校正装置设计完毕后还可以直接从图中读出，不需进行大量烦琐的计算。尤其是在不必精确计算模型参数的场合，只需在图中大致确定变化极点的位置及增益，即可获得校正装置的模型，非常适合于工程设计。

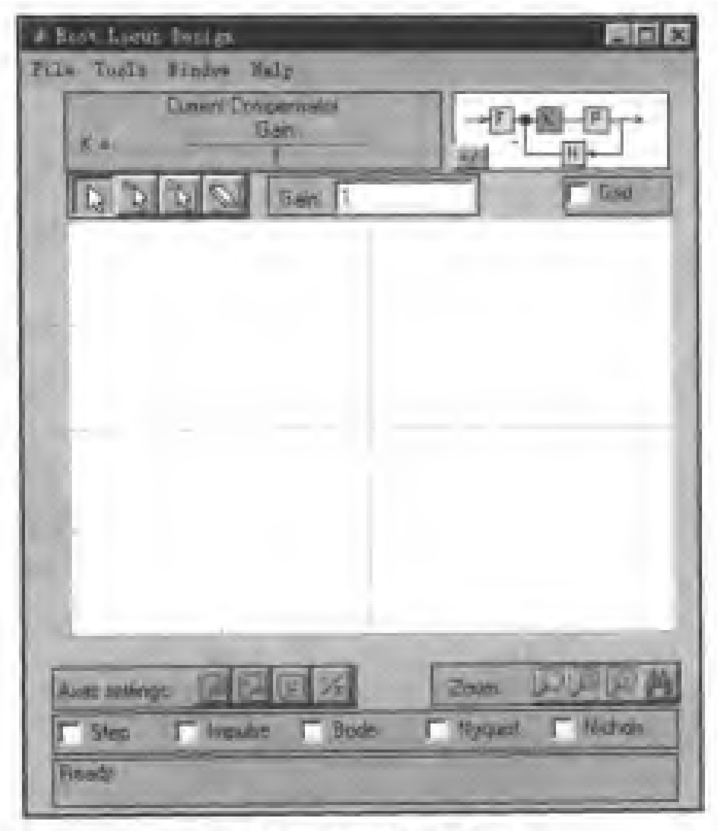


图 10-27　Rltool 仿真界面

选择图 10-27 中“File”菜单的“Import Model...”选项，即可进入如图 10-28 所示的 LTI 系统模型设计界面，输入系统的描述模型。图 10-28 右上角的方框图是系统的零极点传递函数描述模型，其中 F 和 H 是输入和反馈环节，没有特殊要求的话一般设为 1；K 是需求解的根轨迹校正装置，不在此界面输入；P 是被控对象的传递函数，本界面下可选中图 10-28 中间方框“Workspace Contents”下列表的系统描述，然后选择左边的箭头。

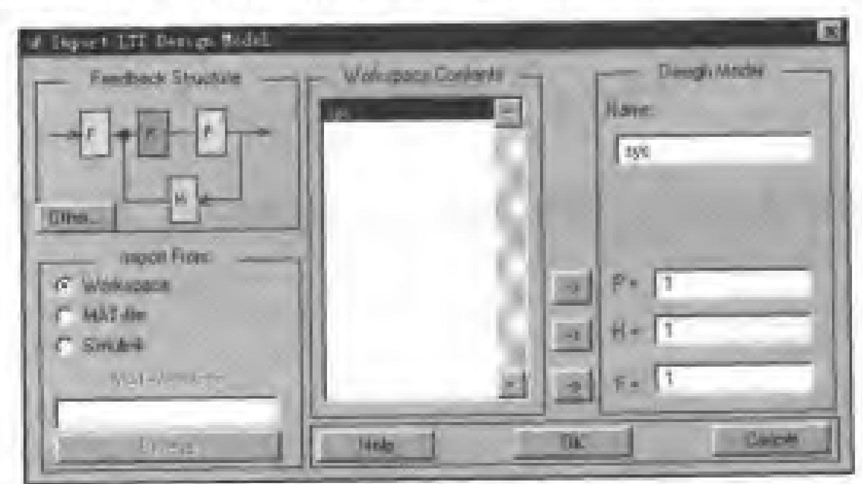


图 10-28　Rltool 环境下系统模型的输入

选择图 10-27 中“File”菜单的“Import Compensator...”选项，即可进入图 10-29 所示的 LTI 系统校正装置输入界面。该界面包括校正装置的名称“Name”对话框、零点选择复选框“Zeros”、极点选择复选框“Poles”以及相关的帮助信息。我们可以通过“Name”对话框赋

予校正装置一个名称，选取下边的 "Add Zero" 按钮和 "Add Pole" 按钮为校正装置增加零极点。



图 10-29　Rltool 环境下校正装置模型的输入

校正装置的输入方式与图 10-28 所示的 LTI 系统模型设计界面相同，但一般校正装置的传递函数都是先使用缺省的 Gain/1，在 Rltool 仿真界面下不断绘制其根轨迹图，通过根轨迹的分布情况和趋势逐步的调整确定最佳的参数。

当然，也可以在 MATLAB Command Window 下设计好初步的校正装置传递函数进行输入（其方法和以前介绍的生成系统描述的方法相同，直接输入传递函数分子和分母多项式即可），再逐步调整。

直接用鼠标点击图 10-27 左上角的系统方框图描述的对应环节，也可以输入相应的传递函数。不过这就是手工编辑，而不是从 "Workspace contents" 中输入了。例如点击 "K" 环节，可以进入图 10-30 的校正装置编辑界面。



图 10-30　Rltool 校正装置编辑界面

该校正装置的模型是以零极点传递函数的形式描述的。修改 "Name" 选项可以更改校正装置的名称，对于大型的控制系统设计来说，这是十分必要的。选择 "Add Zero" 或 "Add Pole" 按钮可以增加校正装置的零点或极点，增加零点的界面如图 10-31 所示。

如果选中图 10-31 的 "Delete" 复选框，则表示删除该零点。增加极点的界面和选择方式与图 10-31 相同，这里就不再赘述了。

如果待校正的被控对象不是非常复杂，控制精度要求不高，可以考虑使用图 10-27 右上角第二行的四个按钮来设置系统的闭环极点和系统的开环零极点位置。



图 10-31 增加校正装置的零点

图 10-27 按下的缺省的按钮的功能是选择校正装置的增益。系统模型输入完毕画出闭环根轨迹后按下该按钮拖动鼠标在图 10-27 下方的坐标图中移动，图 10-27 第二行中间的“Gain”文本框的数值就会发生变化，找到合适的闭环极点（注意，是闭环极点，而非开环极点）位置后单击鼠标左键，即可完成闭环极点设置，同时“Gain”文本框中的数值就是校正装置的增益。

相应的，按钮和按钮的功能就是设置系统开环极点和零点位置的按钮，其使用方法与设置系统闭环极点的按钮相同。不过在实际设计根轨迹的工程中，一般是先确定校正后系统的零极点，然后在画好的根轨迹上确定闭环的极点位置，而不在根轨迹上的点是不能作为闭环极点的。

按钮的功能是取消设置完的系统开环零极点，只需选中该按钮后用鼠标在需要取消的位置上单击即可。

图 10-27 下方还有几个工具条，其中“Axes Settings”是用鼠标设置坐标轴，不过该选项的功能可由“Tool ”菜单下“Set Axes Preference”选项完成，如图 10-32 所示。



图 10-32 根轨迹坐标轴设置

各选项的意义都比较明显，只是其中“Axis Equal”选项表示横坐标和纵坐标的刻度相等，“Axis Square”选项表示不管坐标刻度如何，根轨迹图均为正方形。



图 10-27 中的 [Zoom 工具条] 工具条表示以不同的比例和可视区域显示根轨迹坐标图。各选项从左至右依次是“Zoom in X-Y”、“Zoom in X”、“Zoom in Y”和“Full View”。其中按钮代表的“Full View”选项可以提供根轨迹的全景，便于设计者从总体上把握根轨迹的变化趋势。

图 10-27 还有一个工具条 [Step Impulse Bode Nyquist Nichols]，其功能是分别绘制闭环系统的阶跃响应曲线（Step）、脉冲响应曲线（Impulse）、Bode 图（Bode）、Nyquist 图（Nyquist）和 Nichols 图（Nichols）。这个工具条深得控制系统分析及设计人员的喜爱。有了它，我们就不必将求得的校正装置和原系统模型重新在 MATLAB Command Window 下仿真，根据 plot（）语句或 lsim（）函数绘制这些图形，进行时间域或频率域指标的校验了。

10.4.2　根轨迹校正举例

按根轨迹校正反馈系统可以依以下步骤来进行：

(1) 根据给定的动态性能指标，确定希望的闭环极点的位置。若是二阶系统，直接求出其极点；高阶系统则求其主导极点。

(2) 绘制未校正系统的根轨迹。若希望的主导极点不在此根轨迹上，而且可以看出此根轨迹不可能提供满意的动态性能，那么就说明不能只靠改变增益 K 来满足系统的动态性能要求，必须使用校正装置来改造闭环系统的根轨迹，使其通过希望的主导极点。在手工或 MATLAB Command Window 环境下，需要根据前边讨论的规则设计相应的校正装置，使其满足要求。而 Rltool 仿真环境提供了一些可视化操作手段，不必进行繁琐的计算，用鼠标即可将其配置到满意的位置。

(3) 若校正后（或校正前的）根轨迹已经通过希望的闭环主导极点，则应检验相应的开环比例系数是否满足要求，如不满足，可采用在原点附近增加开环偶极子的办法来提高开环比例系数，同时保持根轨迹仍通过希望的主导极点。

(4) 校核系统的闭环静态和动态指标。

请看下面这个例子：

设某二阶被控系统的传递函数 $G(s)$ 如下。试选用合适的方法设计一个串联校正装置 $K(s)$，使得系统的阶跃响应曲线超调量 $\sigma\%<30\%$，过渡过程时间 $t_s<1.5\text{s}$，开环比例系数 $K>10/\text{s}$。

$$G(s)=\frac{K}{s(s+2)}$$

结果：初步求得的串联校正装置的传递函数为：

$$G(s)=\frac{49.6(s+3.9)}{(s+9.4)}$$

加入偶极子后系统的校正装置传递函数为：

$$G(s)=\frac{49.6(s+3.9)(s+0.12)}{(s+9.4)(s+0.02)}$$

分析：对于本例这种二阶系统，系统闭环阶跃响应的超调量 $\sigma\%$ 和过渡过程时间 t_s 与其传递函数的系数有着确定的关系。即便加入校正装置后成为了三阶或四阶系统，其主导极点和超调量 $\sigma\%$ 和过渡过程时间 t_s 之间同样有着近似的函数关系。因此，通过配置系统闭环极

点的根轨迹校正的方法无疑是解决本例的最佳思路。另外，本例系统的阶次比较低，根轨迹的出发点、会合点和分支数都比较少，便于应用 MATLAB 提供的 Rltool 环境进行直观的设计和求解。

求解过程：

本例的求解分为以下几个部分：

1．输入被控系统并绘制其根轨迹

在 MATLAB Command Window 下键入以下代码：

```
%系统传递函数
num=1;
den=[1 2 0];
%生成抽象的系统描述
sys=tf(num,den);
%调用 rltool 仿真环境
rltool(sys)
```

其中 num 和 den 分别是待校正的系统传递函数模型的分子和分母多项式系数。tf（）是 MATLAB 提供的生产系统传递函数模型抽象描述的函数。键入回车，则进入如图 10-27 所示的 Rltool 根轨迹设计及仿真界面。选择该界面的“File”菜单的“Import Model...”选项，进入如图 10-33 所示被控对象模型输入界面。

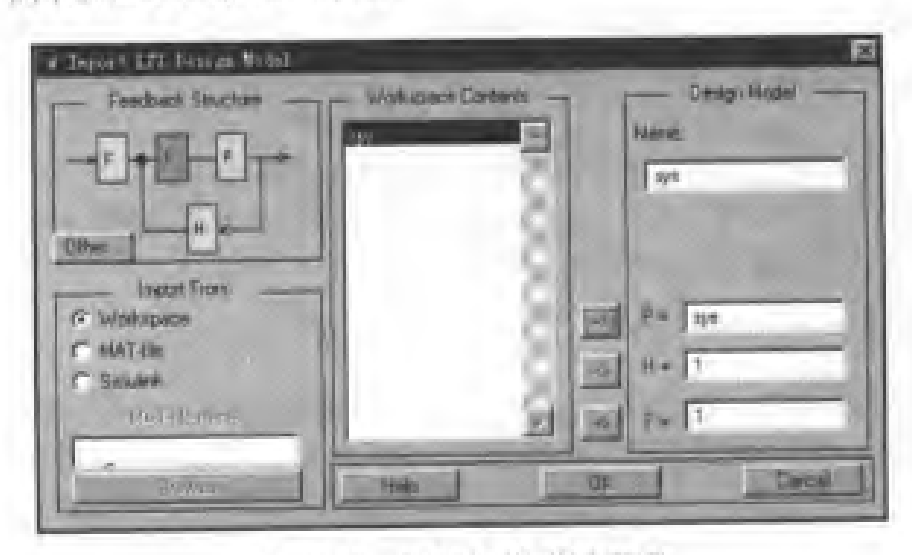


图 10-33　被控对象模型输入界面

选中图中“Workspace Contents”下列表的“sys”选项，然后单击“P”选项旁边的箭头，即可完成被控系统模型的输入。其功能是将刚才在 MATLAB Command Window 下编辑的系统 sys 的传递函数输入到环节“P”中。“P”下边的“H”和“F”分别表示反馈环节和输入环节的传递函数，本例另 H=1，F=1 表示单位负反馈，参考输入为 1。决定后选取“OK”按钮即可。

2．根据系统根轨迹和阶跃响应评价其动态性能，并计算希望的闭环主导极点位置

选取图 10-32 模型输入界面“OK”按钮后即返回入图 10-27 所示的 Rltool 根轨迹设计及仿真界面，此时已绘制完成系统的根轨迹，如图 10-34 所示。

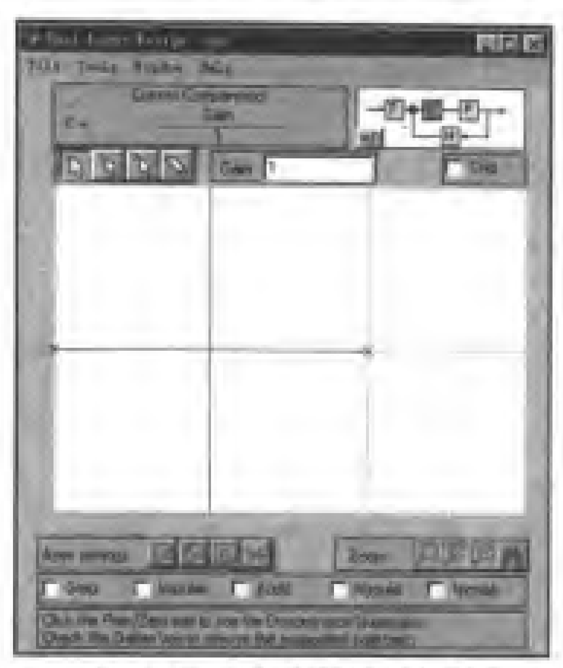


图 10-34　校正前系统的根轨迹

从图中可以看出，无论系统的闭环极点在其根轨迹上怎样变化，与原点的距离都比较接近。这样的系统过渡过程时间较长，很难满足 t_s<1.5s 的要求。选中图 10-34 的 "Step" 选项，可得系统的阶跃响应，如图 10-35 所示。

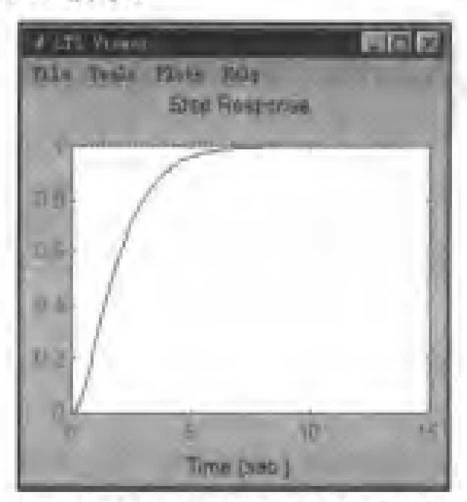


图 10-35　校正前系统的阶跃响应

果然，从图 10-35 中读出，系统的过渡过程时间超过 5s，并且曲线的形状是过阻尼型，快速性很差。

根据系统的动态指标超调量 $\sigma\%<30\%$，过渡过程时间 t_s<1.5s，以及二阶系统极点与动态指标的关系：

$$\sigma = e^{-(\varsigma/\sqrt{1-\varsigma^2})\pi}$$

$$t_s = \frac{4T}{\varsigma}$$

$$p_1、p_2 = -\frac{\varsigma}{T} \pm j\frac{\sqrt{1-\varsigma^2}}{T}$$

可以求得希望的系统闭环主导极点为

$$p_1, p_2 = -3 \pm j3\sqrt{3} = -3 \pm j5.1962$$

数值计算都可以由相应的MATLAB数学运算函数实现，在第1章里已经有详细的介绍，这里就略去了。

3. 计算开环零极点位置并输入

从图10-34可以看出，系统校正前的根轨迹不过希望的闭环极点，因此，必须设计一校正装置，将系统的根轨迹向左弯曲。一般校正装置的传递函数为：

$$K(s) = \frac{s-z}{s-p}$$

具体的设计方法有两种：一种是根据3.3节绘制根轨迹的模条件和角条件求出相应零点z和极点p；另一种是根据一定的规则在Rltool环境下不断试验，直到选出满意的参数。对于本例来说，希望系统的根轨迹向左弯曲，一般是在负实轴上设置一个零点和一个极点，并且零点在极点的左边。弯曲程度随着零极点间距离的增大而增大（这些经验的具体论述在有关根轨迹的控制理论书籍中都可以找到，注意，是经验，而非严格的定理）。本例直接给出结果见图10-36，零点z= -3.9，极点p= -9.4。考虑到物理可实现性，选择的位置都落在实轴上，虚部为零。



图10-36 输入校正装置的零极点

4. 坐标轴设置及校正后根轨迹绘制

从图10-34来看，根轨迹所在的坐标系范围过小，无法直接在原图的左边平面上选择校正装置零极点的位置。因此，首先要修改坐标范围，使之能够显示大范围的根轨迹图。

在图10-33界面下，选择"Tool"菜单下"Set Axes Preference"选项，坐标系修改界面见图10-37。

将横坐标"X-axis"和纵坐标"Y-axis"的下限值均设为-10。由于Rltool仿真环境下坐标系的选择都是对称的，因此上限值可以不必手工设置，Rltool仿真环境可以自动将其与下限值对应起来。当然，也可以只修改上限值，不修改下限值。因为根轨迹关于实轴对称，因此只设定坐标轴的下限或上限即可。至于中间的两个复选框"Axis Equal"和"Axis Square"，这是选择坐标系的外观的。如果用户希望X轴和Y轴采用同样的刻度，那么就选中"Axis Equal"；如果用户希望不管刻度如何，整个坐标系呈正方形，则选中"Axis Square"。我们这里使用缺省的"Axis Equal"选项。图10-37下边的部分是根轨迹颜色和标记的设置，一般选择缺省设置即可。

回到Rltool仿真界面后选择 [Zoom 工具条图标] 工具条的"Full View"按钮，即可见到类似

图 10-38 的根轨迹图形。不同的是此时见到的图形没有图 10-38 的三个红色小方块和网格。这三个红色小方块是系统闭环极点所在的位置，其设置方法是：选择按钮，在求得的主导极点之一（例如$-3+j5.1962$）位置上单击鼠标左键，即可得到这三个红色小方块。同时“Gain”文本框的数值由 1 变为 49.6。当然，位置稍有偏差，“Gain”文本框的数值也稍有不同，不过基本上在 45～55 之间。至于网格，是因为选中了“Grid”复选框，该网格是按照极坐标绘制的，从中我们可以读出根轨迹相应点的模和幅角。至此，初步设计已经完成，下面是校验性能指标。



图 10-37　根轨迹坐标平面设置

图 10-38　初步校正后系统的根轨迹图

5．根据响应曲线校验系统的性能指标

选择图 10-38 的“Step”和“Bode”复选框，可以得到系统的响应曲线，如图 10-39 所示。

图 10-39 是此时系统的闭环阶跃响应曲线。从图 10-39 中可以读出，系统的超调量 $\sigma\%$约为 25%，过渡过程时间 t_s 不超过 1.2s。满足超调量 $\sigma\%<30\%$，过渡过程时间 $t_s<1.5s$ 的要求。因此，校正后动态指标是合格的。

系统的 Bode 图如图 10-40 所示，因为系统存在一个积分环节 1/s，所以其开环增益从第一个非零极点所在的位置 $\omega=2$ 处读取，约为 20dB，即 10/s 左右。本例要求是开环增益 $K>10/s$，因此，该校正装置在这个指标的表现上并不出色。如果投入实际应用，很可能造成不必要的麻烦。

为了解决这个问题，还应该在原来的基础上重新设计一个校正装置，使其满足静态开环增益 $K>10/s$ 的要求。这时有的读者可能会说，把原校正装置的 $K(s)$ 增益“Gain”增大不就可以了吗？很可惜，这种简单的方法是不能成立的。从前边的分析我们也可以看出，改变增益“Gain”后系统闭环极点的位置势必也要发生变化，这样一来，系统的静态性能固然可以满足目标，动态性能就无法保证了。还得重新设计。能不能找到一种校正装置，既可以改变系统的静态开环增益，又不影响其动态性能？答案是肯定的，这就是下一步要用到的偶极子。



图 10-39　初步校正后系统的阶跃响应曲线



图 10-40　初步校正后系统的 Bode 图

6. 增大开环增益——偶极子设计

所谓偶极子，是指复平面上距离很近的一对零点和极点。我们已经知道，有 n-r 个非零极点的系统的开环增益为：

$$K = K'\frac{\prod_{i=1}^{m}(-z_i)}{\prod_{j=1}^{n-r}p_j}$$

如果在原点附近加入一对偶极子 z 和 p，并且令 z=10p，系统的开环增益就变为：

$$K = K'\frac{\prod_{i=1}^{m}(-z_i)}{\prod_{j=1}^{n-r}p_j}\cdot\frac{(-z)}{(-p)} = 10K$$

因此，增加在原点附近的闭环偶极子可以改变系统的开环增益。并且，如果系统的主导极点远离这对偶极子，系统的根轨迹就不会发生太大变化。因为根轨迹开始于极点截止于零点，这对偶极子自己形成了一段根轨迹，不会影响到其他根轨迹。因此，系统的动态性能也不会因为这对偶极子而产生明显的变化。

这就回答了上一步提出的问题，我们找到了一种既可以改变系统的静态开环增益，又不影响其动态性能的校正装置。考虑到系统的变化主导极点为-3±j5.1962，模为 6。根据偶极子的设计原则，应该让偶极子中模较大的一个与系统主导极点的模相差 50 倍以上，并且使偶极子的零点和极点的模相差一定的倍数。因此，我们设计一对偶极子-0.12 和-0.02，将校正装置变为：

$$G(s) = \frac{49.6(s+3.9)(s+0.12)}{(s+9.4)(s+0.02)}$$

为了在 Rltool 仿真界面下配置这对偶极子，首先要点击工具条的“Zoom in X”按钮，将横坐标轴放大到可以辨认的程度，然后在分别用和按钮选择合适的位置放置该偶极子（当然可以从“File”菜单下的“Import Compensator”选项里输入，也可以用

"Tool"菜单下的"Edit Compensator"选项设置)，用删除。偶极子配置结果如下：

从图 10-41 中可以看到，圆圈代表的零点在-0.12 附近，叉号代表的极点在-0.02 附近，它们之间有一条根轨迹，这也符合根轨迹起于极点终于零点的规律。此外，设置好零极点以后，上边校正装置 K 也变成了希望的形式。

从图 10-41 里还可以看到，系统根轨迹的横坐标变成了从-0.2 到 0.05，而纵坐标不变。这样一来，配置该校正装置的偶极子和观察此范围内的根轨迹就变得比较方便了。这段根轨迹之间有一个红色小方块，这是一个闭环极点。不过它离主导极点很远，对系统的动态性能没有什么太大的影响。

如果希望看到此时根轨迹整体的情况和变化，可以点击按钮，得到此时系统的根轨迹的全景图如图 10-42 所示。



图 10-41　偶极子配置

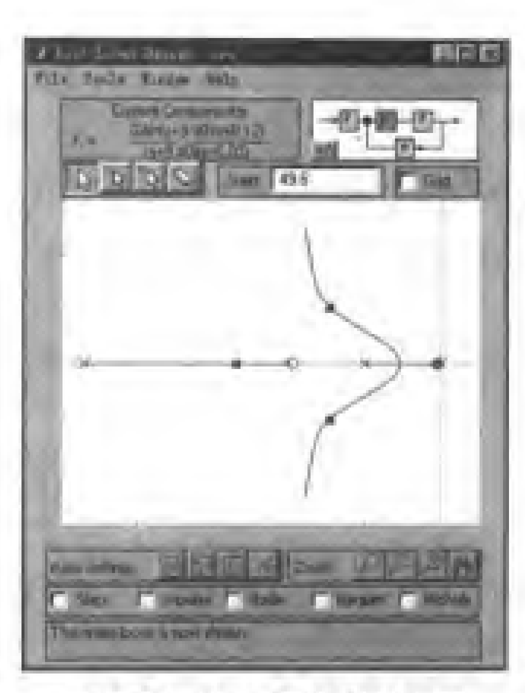


图 10-42　校正后系统的根轨迹图

从图 10-42 中可以看出，系统的根轨迹的形状没有什么太大的变化，尤其是闭环主导极点所在的两条根轨迹仍然是向左弯曲的两条曲线。而闭环主导极点的位置也没有什么变化，依然在-3±j5.2 左右。

可以预见，系统的动态性能与上一步使用不含偶极子的校正装置校正时的动态性能不会有明显的变化。因此，使用此校正装置的系统的动态性能是可以满足要求的。而根据偶极子的特性，此校正装置在低频段会提供大约 6（0.12÷0.02=6）倍于原系统增益的静态开环增益，使得系统的静态性能也可以满足最初的要求。因此，加入偶极子可以在不影响系统动态性能的情况下改善系统的静态性能。

图 10-42 的原点附近有一块根轨迹比较密集的区域，其放大图就是图 10-41。用"Zoom in X"和"Full View"按钮可以实现两图之间的相互转换。

同样，在图 10-42 中选中"Step"和"Bode"复选框，可以得到此时系统的响应曲线，校验最终的设计结果。

系统的阶跃响应曲线如图 10-43 所示。

可以看出，系统的阶跃响应曲线属于跟踪性能较好的振荡型曲线，快速性和稳定性都能满足要求。

系统的 Bode 图如图 10-44 所示。



图 10-43 校正后系统的阶跃响应曲线

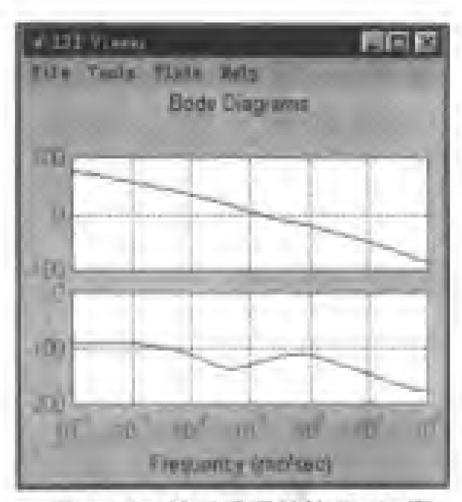


图 10-44 校正后系统的 Bode 图

从图中可以读出，此时系统的超调量 $\sigma\%$ 约为 25%，过渡过程时间 t_s 不超过 1.2s。满足最初的要求。$\omega=0.02$ 处系统的增益接近 35dB，对应开环增益约为 60/s，与前边分析的数据也是相符的。因此，加入偶极子后系统的动态指标和静态指标都能满足要求，从图 10-42 第二行读出的 K 就是最后的校正装置。

小结：本例详细介绍了 Rltool 仿真环境的使用方法和技巧，第一次接触该环境的控制领域的专业人员，尤其是刚开始学习控制理论的学生，一定会深为其直观、快捷和方便而叹为观止。事实上，MATLAB 5.x 加强了有关图形界面设计的功能，经过一定的学习之后我们自己也可以编制出漂亮的图形仿真环境，不过遗憾的是这不是本书的主题，就不再介绍了。感兴趣的读者可以参看 MATLAB 自带的 “GUI Layout”。

本章我们主要介绍了控制系统的校正指标和经验公式、控制系统开环频率特性设计、控制系统的 PID 校正和串联校正在 MATLAB 下的实现方法，最后我们介绍了一个 MATLAB 环境下的根轨迹设计工具——Rltool 仿真环境。Rltool 仿真环境充分体现了 MATLAB 的精髓，即操作完全可视化。因此，作为控制领域的研究者和工程技术人员来说，可以更方便地掌握其使用方法和技巧，把更多的精力放在有关控制系统的设计和仿真等问题上来。

第 11 章　控制系统的状态空间设计方法

从 20 世纪 30 年代以来，基于状态空间模型的控制系统设计方法有了很大的发展。20 世纪 50 年代至 60 年代，控制系统的研究者们提出了各种控制系统设计方法，其中影响较大、比较有代表性的包括极点配置设计方法、多变量系统的解耦控制以及以线性二次型性能（LQR）指标为基础的最优控制问题。

利用状态空间描述和校正线性定常系统的重要特点是校正方法的规范化。校正方法的规范化很适于计算机编程，使得计算机辅助设计成为可能。我们已经知道，在 MATLAB 环境下最基本的元素就是矩阵，这使得使用 MATLAB 进行状态空间模型的综合校正非常方便。关于状态空间的校正，应明确以下几点：

- 校正的目标是什么。
- 采用什么样的控制规律达到这个校正目标。
- 线性定常系统（A，B，C，D）在什么条件下可以实现这个目标。
- 实现校正目标的设计方法是什么。
- 采用该设计方法所得的结果是否惟一。

应当指出的是，状态空间设计方法和频率域的设计方法并不是相互独立地发展的，它们之间有着千丝万缕的联系。对于同一个控制系统来说，可以采用传递函数描述，也可以采用状态空间描述。同样，对于同一个要求的性能指标，可以考虑采用频率域校正方法，也可以考虑状态空间的校正方法。

当然，这两种校正方法各有其优点。一般来说，大型系统的理论推导及证明问题往往需要从状态空间的角度入手，而控制器的设计及校正等问题又要在频率域下实现。而在实际的工程领域，航天技术、精密仪器、机械领域和制造业多采用状态空间设计方法，以化工为代表的流程工业多采用频率域设计方法。

在 MATLAB 环境下，使用状态空间设计方法和使用频率域设计方法都是非常方便的，本章我们主要讨论状态空间设计方法。

11.1　状态反馈与观测

对于控制系统的状态空间描述（A，B，C，D），我们已经详细讨论了它在 MATLAB 下

的实现方式和可控性、可观性的判断。下一步，就是利用状态反馈矩阵对其进行综合校正。对于状态完全可控的系统，我们可以设计反馈校正装置 R 实现闭环极点的任意配置；而对于状态不完全可观的系统，就存在状态重构或称观测器设计的问题。本节将详细讨论这些问题并给出 MATLAB 实现。

11.1.1 极点配置

控制系统的各种特性及其各种品质指标很大程度上由其闭环系统的零点和极点的位置决定。极点配置问题就是通过对状态反馈矩阵的选择，使闭环系统的极点配置在所希望的位置上，从而达到一定的性能指标要求。

希望极点组的选择是一个确定校正目标问题。一般说，这是一个复杂的问题，是一个工程实践与理论相结合的问题。需要注意以下几条：

- 对于 n 维系统，应当指定且只应当指定 n 个希望的极点。
- 希望的极点可以是实数，也可以是按共轭对出现的复数。
- 确定希望极点的位置，需要考虑极点和零点在复数平面上的分布，从工程实际出发加以解决。

对于 SISO 系统来说，其极点配置问题一般可参照下面的思路：

设控制系统的状态方程模型为（A，B，C，D），当引入状态反馈后，系统的控制信号为 $u=r-K^TX$，这里 r 是系统外部的输入；K 是一列向量，有时也称为反馈矩阵；此时闭环系统状态方程模型为

$$\begin{cases}\dot{X}=(A-BK^T)X+Br\\ Y=(C-DK^T)X+Dr\end{cases}$$

可以证明，当系统状态完全可控时，则可以通过状态反馈将系统的极点配置在复平面的任何位置上。

这里给出常用的两种配置系统闭环极点的方法以及相应的 MATLAB 实现，具体的证明请参阅相关书籍。

1．Gura-Bass 算法

假定期望的闭环极点为 $p_i, i=1,2,\cdots,n$，则原系统的开环特征方程 $\alpha(s)$ 和闭环系统特征方程 $\beta(s)$ 分别可以写成

$$\alpha(s)=\det(sI-A)$$
$$=s^n+\sum_{i=0}^{n-1}\alpha_i s^i$$

$$\beta(s)=\prod_{i=1}^{n}(s-p_i)$$
$$=s^n+\sum_{i=0}^{n-1}\beta_i s^i$$

若原系统状态完全可控，则状态反馈矩阵 K 可由下式求出

$$K^T=(\alpha-\beta)^T L^{-1}Q^{-1}$$

式中各参量的含义为

$$(\alpha-\beta)^T=((\alpha_0-\beta_0)\quad,\quad\cdots\quad,\quad(\alpha_0-\beta_0)),$$

$$L=\begin{pmatrix}\alpha_1 & \alpha_1 & \cdots & \alpha_1 & 1\\ \alpha_1 & \alpha_1 & \cdots & 1 & \\ \vdots & \vdots & 1 & & \\ \alpha_1 & 1 & & & \\ 1 & & & & \end{pmatrix}$$

$$Q=\begin{pmatrix}B & , & AB & , & \cdots & , & A^{n-1}B\end{pmatrix}$$

可见，若原系统状态完全可控，则 Q 的逆矩阵存在。而 L 是 Hankel 矩阵，因此总能解出状态反馈矩阵 K，因此能够将系统的闭环极点配置在复平面的任何一个位置。

对于 Gura-Bass 算法，MATLAB 控制系统工具箱中并没有给出相应的可以一步求解状态反馈矩阵的函数。不过从上面的分析中我们可以看出，状态反馈矩阵的表达式很简单，都是矩阵运算。而且细心的读者可能还记得，在第 1 章里我们曾经介绍过 MATLAB 环境下如何生成 Hankel 矩阵的方法。因此，我们可以自己编写用 Gura-Bass 算法求解状态反馈矩阵的 M 文件。

2．Ackermann 配置算法

设问题的描述与 Gura-Bass 算法相同。因为系统状态完全可控，所以 Ackermann 配置算法是先把系统的描述化为某种可控规范型，然后利用规范型的性质求解。具体的证明这里就不讨论了，只给出最后的结果：

$$K^T=(0,\mathrm{L}\ ,0,1)Q^{-1}\beta(A)$$

各参量的含义与 Gura-Bass 算法相同，K 是所求的状态反馈矩阵。

MATLAB 提供了一条基于 Ackermann 配置算法求解系统状态反馈矩阵的函数 acker（），其基本调用格式为

K = ACKER(A,B,P)

其中 A，B 就是系统状态方程描述的参数矩阵；P 是期望的极点向量；K 就是所求的状态反馈矩阵。不过 MATLAB 同时警告说，该算法的数值稳定性并不是非常好，而且当系统的阶次高于 10 或系统的可控性矩阵接近奇异时，有可能引起算法的崩溃。不过 MATLAB 同时还指出，如果求得的闭环非零极点与期望的极点向量 P 的规定值相差超过 10%的话，MATLAB 会提出警告。

一般说来，在 MATLAB 环境下，无论是应用 Gura-Bass 算法手工计算状态反馈矩阵 K 还是应用 acker（）函数计算，首先都应该对相同的状态可控性作出判断。否则就有可能得出错误的结果。尤其是在设计大型控制系统时，虽然 MATLAB 会提出警告，但最好还是掌握每一步计算的流程。

设某小球—滑板系统如图 11-1 所示。

通过改变齿轮的角度 θ 可以达到控制小球的位置 r 的目的。其状态方程如下，其中小球质量 M =0.11 kg，小球直径 R=0.015 m，杠杆偏移量 d=0.03 m，重力加速度 g=9.8 m/s^2，滑板长度 L=1.0 m，小球的转动惯量 J=9.99e-6 kgm^2，r 是小球的位置坐标，alpha 是滑板的角度，theta 是齿轮的角度。四个状态变量分别是小球位置 r、r 的一阶导数、滑板的倾角 alpha、

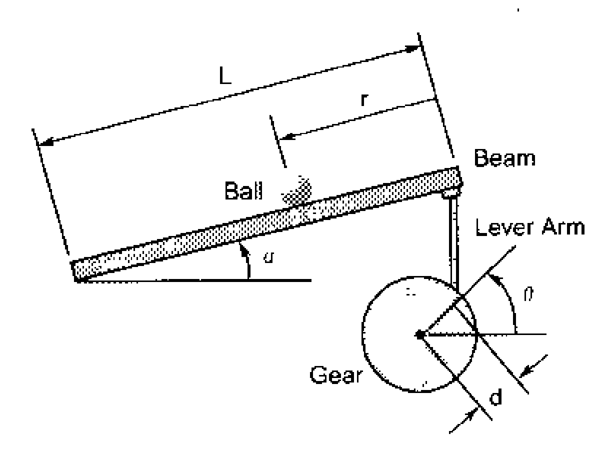


图 11-1　小球—滑板控制系统示意图

alpha 的一阶导数。控制量 u 是齿轮的角度 θ，输出量是小球的位置 r。试根据其状态方程评价其运动特性，并分别根据 Gura-bass 算法和 Ackermann 算法配置其闭环极点，使其阶跃响应满足：过渡过程时间 ts<3s，超调量 $\sigma\%<5\%$。

$$A=\begin{pmatrix} & 1 & & \\ & & \dfrac{-mg}{(\dfrac{J}{r^2}+m)} & \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad B=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$C=\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix} \qquad D=0$$

结果：未校正前系统的零极点和增益为

z =　Empty matrix: 0-by-1

p =　0　0　0　0

k =　7.0000

校正后系统的零极点和增益为

z =　Empty matrix: 0-by-1

p =　−80.0000　　−20.0000　　−2.0000 + 2.0000i　　−2.0000 − 2.0000i

k =　7.0000

状态反馈矩阵为

K =　1.0e+003 *

1.8286　　1.0286　　2.0080　　0.1040

分析：一般来说，对于这种机械控制系统，不加反馈的开环控制很难获得满意的效果，必须要增加状态反馈矩阵。因此，本例的关键就在于将系统要求的性能指标转化为复平面上极点的位置。解决这一步后，相应的极点配置问题就可以通过 MATLAB 编程或 acker () 函数来解决。在 MATLAB 环境下，还可以通过绘图来验证校正的结果。当然，在求解之前还要进行可控性的判断。

求解过程：本例的求解分为以下几步：

3. 原控制系统分析及期望性能指标与闭环极点间的转化

首先要分析一下原系统的零极点分布及阶跃响应的情况（图 11-2），看看它和期望的性能指标之间有什么差距。

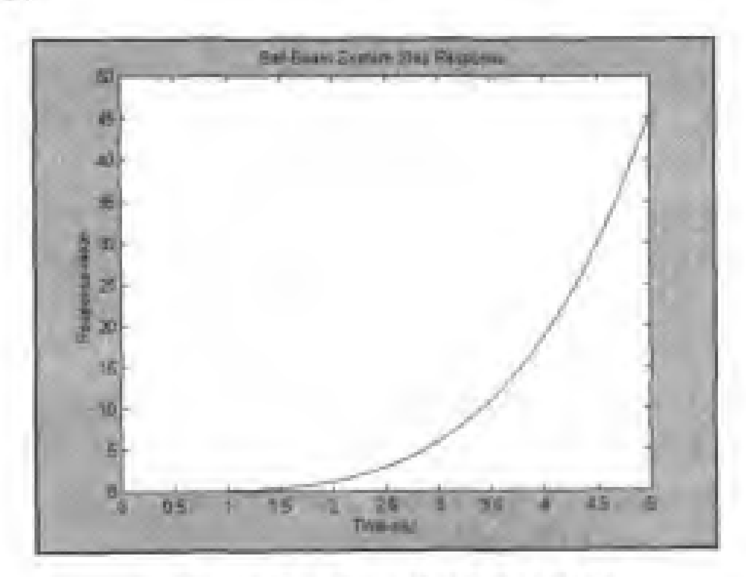


图 11-2 小球—滑板系统校正前阶跃响应

在 MATLAB Editor/Debugger 下输入以下代码：

```
%系统参数初始化
m = 0.111;
R = 0.015;
g = -9.8;
J = 9.99e-6;
H = -m*g/(J/(R^2)+m);
%系统状态方程模型
A=[0 1 0 0
   0 0 H 0
   0 0 0 1
   0 0 0 0];
B=[0;0;0;1];
C=[1 0 0 0];
D=[0];
%原系统的零极点和增益
[z,p,k]=ss2zp(A,B,C,D,1)
%绘制其阶跃响应
T = 0:0.01:5;
U = 0.25*ones(size(T));
[Y,X]=lsim(A,B,C,D,U,T);
plot(T,Y);
```

```
title('Ball-Beam System Step Response');
xlabel('Time-sec');
ylabel('Response-value');
```

校正前系统的零极点和增益可参看上一步“结果”中的数据，可以看到，系统没有零点，极点均为零。这样的系统是非常不稳定的。从图 11-2 系统的阶跃响应曲线来看，在阶跃输入下其响应值趋于无穷大，根本谈不上与期望的性能指标作比较，必须采取状态反馈的校正手段。

然后要将期望的性能指标转化为复平面上极点的位置。这一步在第四章中有类似的例子，我们再复习一下。其思路就是根据经验公式和性能指标确定一对主导闭环极点，然后将非主导极点放在复平面上远离主导极点的地方。在 MATLAB 下求得的主导极点为 $-2\pm j2$，因为原系统是四阶的，所以选取另外两个非主导极点为-20 和-80。

4. 判断可控性并分别按照 Gura-bass 算法和 Ackermann 算法配置其闭环极点

因为系统的四个极点都要配置到新的位置，所以必须对系统的可控性作出判断。只有状态完全可控的系统才能完成这一工作。否则就必须根据原来的物理系统对列出的状态方程作出调整。在上一步的窗口下输入以下代码：

```
%系统的阶次
n=length(A);
%状态可控性矩阵
Q=zeros(n);
Q(:,1)=B;
for i=2:n
%求解状态可控性矩阵
    Q(:,i)=A*Q(:,i-1);
end
%可控性矩阵的阶次
m=rank(Q);
%判断系统是否可控
if m==n
%系统完全可控
    disp('System State Variables can be totally controlled');
else
%系统不完全可控
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
    m
end
```

结果为：

```
System State Variables can be totally controlled
```

因此，系统状态完全可控，可以对系统的四个极点进行任意位置的配置。下面首先按照

Gura-bass 算法编程：

```
%系统的期望闭环极点
p1=-2+2i;
p2=-2-2i;
p3=-20;
p4=-80;
P=[p1,p2,p3,p4];
%Gura-Bass 校正算法
alphaS=poly(A);
betaS=conv([1, -p1],conv([1,-p2],conv([1, -p3],[1, -p4])));
i=length(alphaS): -1:2;
diff=alphaS(i)-betaS(i);
L=hankel([alphaS(length(alphaS) -1: -1 2)';1]);
K=diff*inv(L)*inv(Q);
disp('The state-feedback Matrix is:');
K
disp('The close-loop poles are:');
eig(A-B*K)
```

结果为：

```
The state-feedback Matrix is:
K =   1.0e+003 *
      1.8286      1.0286      2.0080      0.1040
The close-loop poles are:
ans = -80.0000
      -20.0000
      -2.0000 + 2.0000i
      -2.0000 - 2.0000i
```

然后调用 acker（）函数，用 Ackermann 算法配置其闭环极点：

```
K=acker(A,B,P);
```

结果为：

```
K =   1.0e+003 *
      1.8286      1.0286      2.0080      0.1040
```

从以上的结果中可以看出，用 Gura-bass 算法编程和直接使用 acker（）函数得到的结果是完全一样的。不过一般控制理论的入门书籍中讲述的都是 Gura-bass 算法，我们这里给出该算法的源程序也是为了使读者更加清晰的掌握状态反馈校正的设计思路和实现方法。如果是控制领域的工程技术人员的实际应用，如果阶次低于 10 阶的话，还是直接调用 acker（）函数更加简便。

5. 闭环阶跃响应评价

为了检验校正的结果，我们可以绘制其闭环阶跃响应曲线如图 11-3 所示。



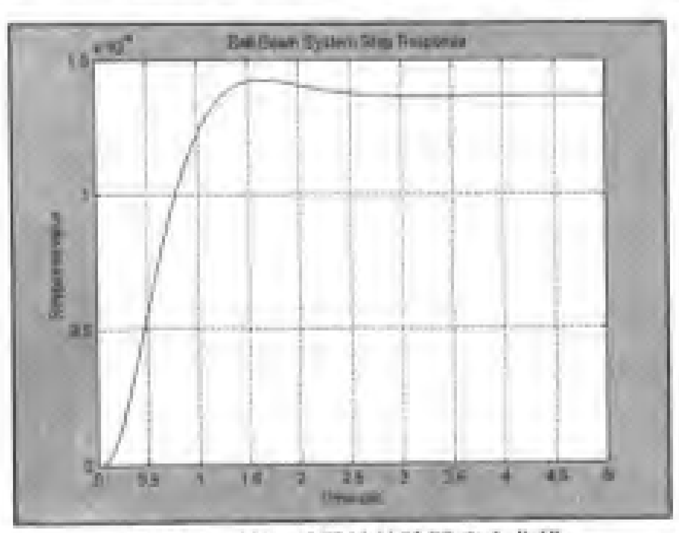


图 11-3　校正后系统的阶跃响应曲线

相应的代码为：

```
T = 0:0.01:5;
U = 0.25*ones(size(T));
[Y,X]=lsim(A-B*K,B,C,D,U,T);
plot(T,Y);
title('Ball-Beam System Step Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
```

从图 11-3 中可以看出，系统的快速性很好，过渡过程时间不超过 2.5s，并且响应过程中只振荡了一次，超调量也非常小，基本满足最初的设计要求过渡过程时间 ts<3s，超调量 σ%<5%。

不过如果仔细阅读了源代码的读者就会发现，系统的输入量是 0.25 倍的阶跃函数，但阶跃响应的稳态值仅为 1.4e-4，相差了三个数量级。出现这种情况的原因是因为全状态反馈自身的特性。与其他反馈手段不同，全状态反馈不但要将输出量反馈回去，还要将所有的状态变量反馈到输入端。而输入端的输入量只有一个，静态无差是指输出量反馈回来后与该输入量差值为零。这么多的状态变量都反馈回来，还要做到差值为零，输出量 Y 势必要发生变化，其减小或增大与状态反馈矩阵 K 有关。

本例中 K 是 10 的三次方数量级的，因此输出量 Y 也就减小相应的倍数。为了达到静态无差，必须在输出端设计一种装置，改善这种状况。

6. 求解参考输入并检验结果

改善这种状况的方法就是将系统的输入函数乘以一个参考量，得到系统的参考输入（Reference Input），以此参考输入与反馈回来的值进行比较得到控制量。其系统结构图如图 11-4 所示。

关于参考输入倍数的计算请参看有关的控制理论书籍。MATLAB 下没有给出相关的函数，但密歇根大学（University of Michigan）的 Yanjie Sun 博士给出了一个 M 函数 rscale（），可以求解系统参考输入。

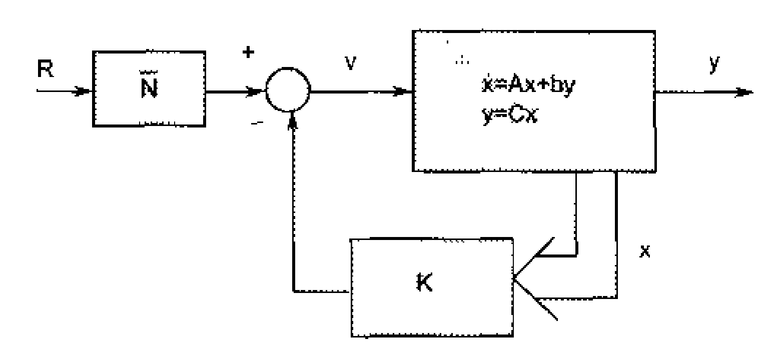


图 11-4　包含参考输入的系统结构图

这个函数属于自由软件，读者可以在相关网站上下载到。下面给出了该函数的源代码，读者可以将下面的代码输入到计算机，保存在 MATLAB 的“BIN”目录下就可以使用了。事实上，MATLAB 的相关网站上有许多可供自由下载的 M 函数，我们在附录中会作出相应的介绍。rscale（）函数代码为：

```
function[Nbar]=rscale(A,B,C,D,K)
% Given the single-input linear system:
%          .
%        x = Ax + Bu
%        y = Cx + Du
% and the feedback matrix K,
%
% the function rscale(A,B,C,D,K) finds the scale factor N which will
% eliminate the steady-state error to a step reference
% using the schematic below:
%
%                          /---------\
%        R          +    u  | .         |
%        ---> N --->() ---->| X=Ax+Bu |--> y=Cx ---> y
%                  -|       \---------/
%                   |              |
%                   |<---- K <----|
%
%8/21/96 Yanjie Sun of the University of Michigan
%          under the supervision of Prof. D. Tilbury
%
s = size(A,1);
Z = [zeros([1,s]) 1];
N = inv([A,B;C,D])*Z';
Nx = N(1:s);
Nu = N(1+s);
```

```
Nbar=Nu + K*Nx;
```

有了这部分代码，就可以计算出参考输入的倍数 Nbar，然后用 Nbar 乘以 B 矩阵就可以得到系统的参考输入。使用参考输入重新求解系统的闭环阶跃响应，就可以让输出量严格跟随输入量的变化了。输入以下代码：

```
Nbar=rscale(A,B,C,D,K);
disp('The Reference Input Value is:');
Nbar
T = 0:0.01:5;
U = 0.25*ones(size(T));
[Y,X]=lsim(A-B*K,B*Nbar,C,D,U,T);
plot(T,Y)
title('Ball-Beam System Step Response with Reference Input');
xlabel('Time-sec');
ylabel('Response-value');
grid;
```

则系统的参考输入倍数为：

```
The Reference Input Value is
Nbar =    1.8286e+003
```

当然，用输入量的稳态值除以输出量的稳态值也能得到这个结果。不过那就是一种意义并不很明确的补偿办法，而不是理论分析得出的结果了。

相应的系统闭环阶跃响应如图 11-5 所示。

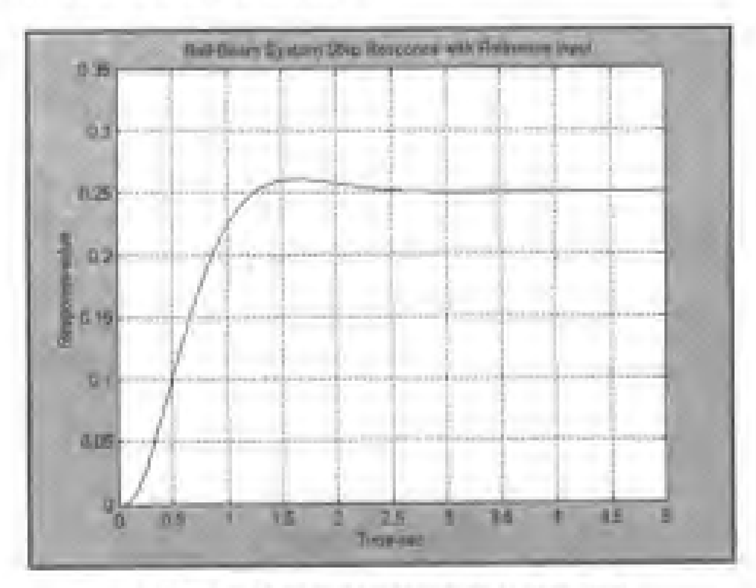


图 11-5　加入参考输入后系统的阶跃响应

从图 11-5 中可以看到，系统的稳态值与阶跃函数的输入值相同，均为 0.25。并且过渡过程时间和超调量与图 11-3 相比均未发生变化，仍旧满足过渡过程时间 t_s<3s，超调量 σ%<5%的要求。这一点很好理解，因为仅仅是输入端乘以了一个常数，系统的结构和闭环极点均未发生变化，主导极点还在原来的位置。因此，系统的动态性能指标也不应该有变化，只是

稳态值提高了三个数量级。

小结：本例详细讨论了在状态空间描述下，使用 Gura-bass 算法和 Ackermann 算法配置系统闭环极点的思路和具体的 MATLAB 实现方法。并且根据实际情况，给出了一个实现参考输入的 M 函数。

从校正的效果来看，系统的各项性能指标还是能够令人满意的。对比上一章的根轨迹校正方法，同样是配置闭环极点，采用根轨迹校正时必须改变系统的结构（增加校正装置）。而配置状态方程的闭环极点则只需设计相应的状态反馈矩阵，状态变量还是原来的四个量，也就是说，系统的结构没有发生改变。对于实际的工程领域来说，有时候这一点是非常重要的。因为有些封装牢固或比较精密的仪器是不允许改变结构的，此时就显示出状态空间极点配置方法的优越性了。

并且，在 MATLAB 环境下，配置状态方程的闭环极点只需一条 acker（）关键语句即可，不必像根轨迹校正或 PID 校正那样绘制大量图形。数学基础和理论完善，计算机仿真软件实现比较简单，这也是以状态方程为基础的现代控制理论蓬勃发展的重要原因。当然，频率域设计也有其优点，特别是对系统的机理了解不是很清楚、控制指标要求不高时，采用频率域的方法还是非常有效的。总而言之，这两种方法互为补充、互相渗透，都是控制系统设计的重要手段。

MATLAB 中还有一条配置状态方程闭环极点的语句 place（），并且它是为 MIMO 系统设计的（acker（）函数是为 SISO 系统设计的）。其基本调用格式与 acker（）函数类似，但该函数有两个可选的返回值 PREC 和 MESSAGE。PREC 表示求得的 A-B*K 的特征值与期望的闭环极点之间的相似程度；如果求得的极点与期望的闭环极点之间相差超过 10%的话，相关的警告信息就存在 MESSAGE 里面。

虽然 place（）函数是为 MIMO 系统设计的，但同样适用于 SISO 系统（为 SISO 系统设计的函数一般不适用于 MIMO 系统）。例如：对于前例，在其工作空间（Workspace）的 MATLAB Command Window 下输入：

```
[K,PREC,MESSAGE] = PLACE(A,B,P)
```

结果为

```
K = 1.0e+003 *
    1.8286    1.0286    2.0080    0.1040
PREC =    15
MESSAGE = ''
```

求得的状态反馈矩阵 K 与 acker（）函数的结果一样。PREC 表示的是求得的极点与期望的闭环极点相同的小数位数，15 是一个很高的数字了，说明两者之间的差别非常小。MESSAGE 里没有信息，说明对于本例的数值，该算法是稳定的。一般说来，PREC 很高，MESSAGE 里就没有信息。而 PREC 较低时，一般是系统的参数矩阵接近奇异或状态不完全可控，此时系统会在 MESSAGE 里给出信息：

```
MESSAGE = 'Matrix A is ill conditioning'
```

表明矩阵 A 为病态矩阵。此时实现全极点状态反馈是不可能的，一般要改变系统的结构，从而改变矩阵 A 的参数。如果 MESSAGE 里给出信息是系统接近不完全可控，则需要重新选择状态变量，使新的状态变量完全可控。

11.1.2 状态观测器

上一小节中我们叙述了状态完全可控的系统（A，B，C，D）可以通过状态反馈任意配置闭环极点。为了实现状态反馈，需要系统所有的状态信息。但是系统的所有状态不一定都能测量到，这就造成了状态反馈在物理实现上的困难。也就是说，即使理论上证明了系统状态完全可控，能实现全极点状态反馈，也必须根据系统的实际情况来作出选择。这就提出了状态重构问题。

状态重构问题的核心，就是重新构造一个系统，利用原系统可以直接测量的变量如输出量 y 和输入量 u 作为它的输入信号，并使其输出信号 x'（t）在一定指标下和原系统的状态变量 x（t）等价。通常把 x'（t）叫做 x（t）的状态重构或状态估计，而把实现状态重构的系统叫做观测器。此时系统结构图如图 11-6 所示。

x'（t）和 x（t）间的等价性指标常常采用渐进等价的指标，即：

$$\lim_{t\to\infty}\tilde{x}(t)=\lim_{t\to\infty}\left[x(t)-x'(t)\right]=0$$

其中 $\tilde{x}(t)=x(t)-x'(t)$ 称为观测误差。

该观测结构的基本方程为：

$$\dot{x}'=(A-LC)x'+Bu+Ly$$

可见，该基本观测器可以任意配置极点的充分必要条件是矩阵对（A，C）完全可观测。完全可观系统的基本观测器的极点配置设计可以仿照完全可控系统用状态反馈进行极点配置的方法进行。请看下例。

某电磁振荡系统如图 11-7 所示。

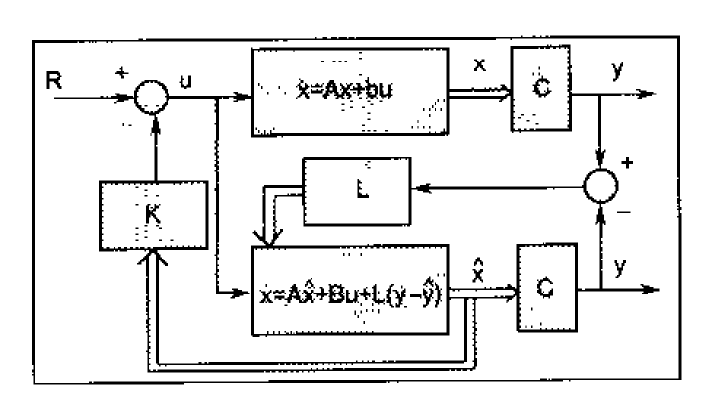


图 11-6 加入观测器后的系统结构图

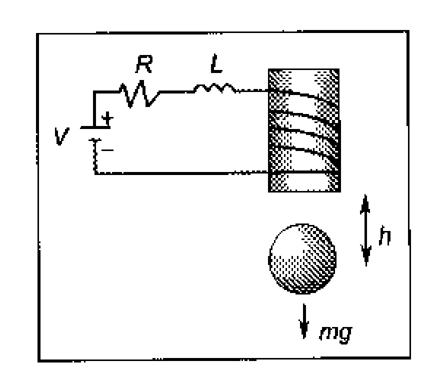


图 11-7 电磁振荡系统示意图

状态方程已知，三个状态变量分别是小球与线圈的位置变化量Δh、Δh 的一阶导数和电流的变化量Δi。输入量是电源电压 V，输出量是Δh。其中小球质量 M=0.05kg ，电磁常数 $K=0.0001$，电感 $L=0.01$ H，电阻 $R=1\,\Omega$，重力加速度 $g=9.81$ m/s^2。

该电磁振荡系统的控制要求为：

(1) 设计状态反馈矩阵 K 及参考输入，满足过渡过程时间 t_s<0.5s，超调量 σ%<15%。

(2) 假设系统的某些状态变量无法直接测量，试选择合适的参数设计状态观测器，给出相应的观测误差。

结果：

求得的状态反馈矩阵为：

K = 　−280.7143　　−7.7857　　−0.3000

观测器矩阵为：

$$A=\begin{pmatrix}0 & 1 & 0\\ 980 & 0 & -2.8\\ 0 & 0 & -100\end{pmatrix}\qquad B=\begin{pmatrix}0\\ 0\\ 100\end{pmatrix}$$

$$C=\begin{pmatrix}1 & 0 & 0\end{pmatrix}\qquad D=0$$

```
L =   1.0e+004 *   0.0203
                   1.1282
                   0
```

本例的解题步骤分为以下几步：

1．原控制系统分析及期望性能指标与闭环极点间的转化

与前例相同，首先要分析原系统的极点分布和阶跃响应的基本情况，判断采用何种校正手段。在 MATLAB Editor/Debugger 下输入以下代码：

```
%系统状态方程描述
A = [  0    1     0
      980   0   -2.8
       0    0   -100];
B = [  0
       0
      100];
C = [1     0     0];
D=   0;
%系统的极点分布情况
poles = eig(A);
%闭环阶跃响应曲线绘制
t = 0:0.01:2;
u = 0.25*ones(size(t));
x0 = [0.005 0 0];
[y,x] = lsim(A,B,C,D,u,t,x0);
h = x(:,2);
plot(t,h);
title('Magnetic System Step Response');
xlabel('Time-sec');
ylabel('Response');
grid;
```

求得系统的闭环极点为

ans =　31.3050

```
-31.3050
-100.0000
```

可以看到，系统有一个右半平面的极点 31.3050，因此未加校正前系统的阶跃响应是不稳定的。从图 11-8 的曲线来看也是如此，曲线以极快的速度最终趋于无穷。考虑到状态方程已知，可用状态反馈的方法将闭环极点配置在任意的位置。

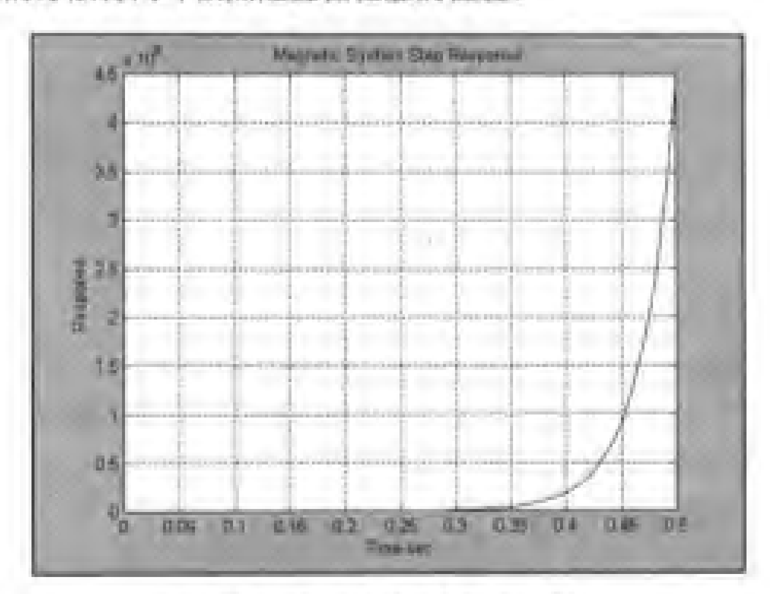


图 11-8　原系统闭环阶跃响应曲线

根据性能指标的要求，过渡过程时间 ts<0.5s，超调量 σ %<15%，求得期望的闭环主导极点为-10±j10（具体求解方法可参看例 5.1 或第四章 4.4 节）。因为系统是三阶的，选取另一个非主导极点为-50。

2．求解相应的状态反馈矩阵及参考输入

通过前例我们知道，想要让输出量跟随输入量的变化必须增加参考输入。因此，首先判断可控性，然后根据期望闭环极点求解状态反馈矩阵，最后求出参考输入。紧接上一步，输入代码：

```
%期望闭环极点
p1 = -10 + 10i;
p2 = -10 - 10i;
p3 = -50;
P=[p1,p2,p3];
%判断可控性
n=length(A);
Q=zeros(n);
Q(:,1)=B;
%求解可控性矩阵
for i=2:n
    Q(:,i)=A*Q(:,i-1);
end
```




```
m=rank(Q);
if m==n
    K=place(A,B,P);
else
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
    m
end
%求解参考输入
Nbar=rscale(A,B,C,D,K);
lsim(A-B*K,B*Nbar,C,D,u,t);
grid;
```

求得的校正后系统的阶跃响应曲线如图 11-9 所示。

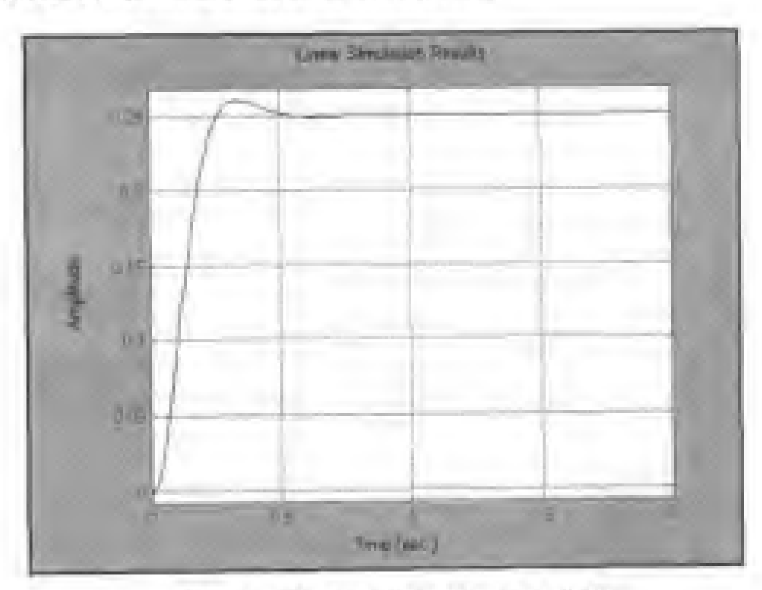


图 11-9　校正后系统的阶跃响应曲线

相应的状态反馈矩阵为

K =　−280.7143　−7.7857　−0.3000

参考输入倍数为

Nbar =　−35.7143

调用 place（）函数返回的精度为

place: ndigits= 15

表明配置的极点与期望闭环极点之间的差距非常小。从图 11-9 的阶跃响应曲线来看，系统的快速性和稳定性都满足最初要求的过渡过程时间 t_s<0.5s，超调量 σ%<15%（注意，这里为了观察得更清楚，输入量选用的是 0.25 倍的阶跃函数）。并且，系统在达到稳态值之前只振荡了一次。

3．求解状态观测器

对于实际的控制系统来说，即使状态完全可控，也很难做到系统的每个状态变量都能够直



接用仪器测量。因此，有必要设计系统的状态观测器，通过重构的等价状态变量来构成状态反馈。一般说来，如果观测器构造得比较好，使用重构的等价状态变量进行状态反馈和使用原状态变量进行状态反馈没有什么太大的区别。因此，我们用观测误差曲线来评价观测器的性能。

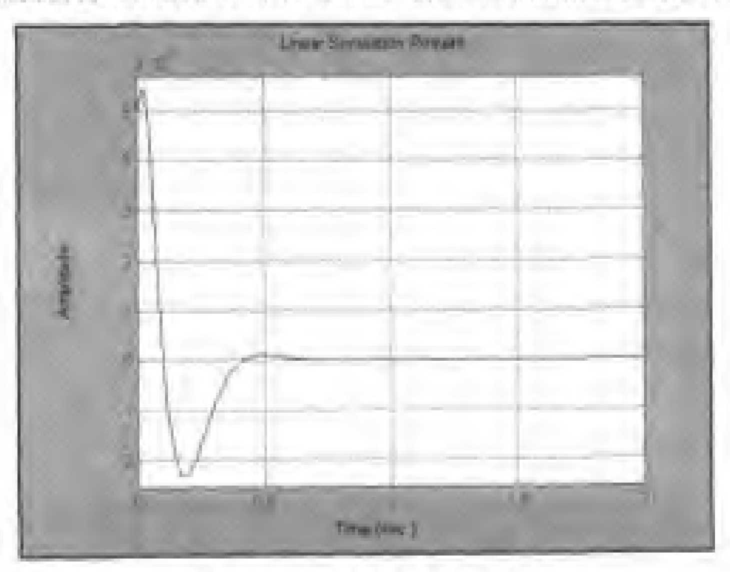


图 11-10　观测器误差响应曲线

图 11-10 为系统的观测器误差曲线，其纵坐标的数量级为 10^{-3}，表明误差在一个很小的范围内振荡，观测器能够反映状态变量的变化。

相应的状态变量的观测值响应曲线见图 11-11，由于观测误差的数量级为 10^{-3}，因此可以认为重构的状态变量能够反映原状态变量的变化。

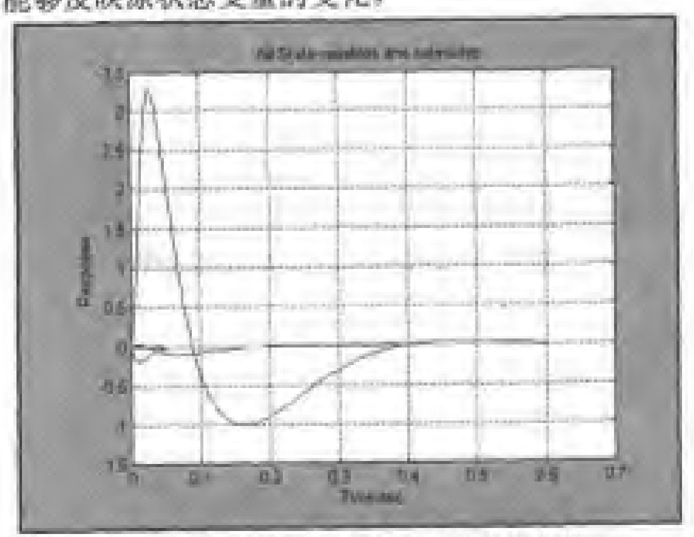


图 11-11　各状态变量的观测值响应曲线

在求解观测器时，首先要判断系统的可观性，然后配置观测器的极点。如果系统完全可观的话，我们希望观测器的响应要快于原系统的响应。因此，将观测器的三个极点均配置在 -100 左右。具体的数字可以自己选择，其原则是在实际系统允许的情况下尽可能的靠近复平面的左边。紧接上一步，输入以下代码：

```
%系统观测器极点
op1 = -100;
```



```
op2 = -101;
op3 = -102;
%求解系统可观性矩阵
n=length(A);
Q=zeros(n);
Q(1,:)=C;
for i=2:n
    Q(i,:)=Q(i-1,:)*A;
end
m=rank(Q);
if m==n
%求解观测器增益矩阵
L = place(A',C',[op1 op2 op3])';
%重构的系统及状态变量
At = [A - B*K              B*K
        zeros(size(A))       A - L*C];
Bt = [      B*Nbar
          zeros(size(B))];
Ct = [     C           zeros(size(C))];
%输出量观测误差
lsim(At,Bt,Ct,D,zeros(size(t)),t,[x0 x0])
grid;
%各状态变量观测误差
t=0:0.005:0.6;
[e,x]=lsim(At,Bt,Ct,D,zeros(size(t)),t,[x0 x0])
plot(t,x);
title('All State-variables and estimates');
xlabel('Time-sec');
ylabel('Response');
grid;
else
    disp('System State Variables cannot be totally Observed');
    disp('The rank of System Observable Matrix is:');
    m
end
```

求得的观测器增益矩阵为

```
L =   1.0e+004 *   0.0203
                   1.1282
                   0
```

相应的输出量观测误差和各状态变量观测值参看图 11-10 和图 11-11。从这两幅图中可以看出，虽然误差响应有一定的振荡，但在 0.5s 之前都已经基本回到了零点。也就是说，0.5s 以后重构的等价状态变量，已经可以代替原来的状态变量实现全状态反馈了。此时再根据第 2 步设计的状态反馈矩阵 K 进行运算，就可以把系统的闭环极点配置在复平面上希望的位置了。

小结：本例一方面复习了状态反馈矩阵的设计方法，另一方面又着重介绍了状态观测器的设计和 MATLAB 实现。从控制的效果来看，还是比较满意的。在控制理论的实际应用中，观测器的设计是非常重要的环节。一方面如前所述，系统的状态变量无法完全直接测量；另一方面有的系统的状态方程就是由传递函数转化得到的，其状态变量没有对应的物理意义，只能构造状态观测器来实现状态反馈。当然，这样做的前提是，系统的状态变量既是完全可控的又是完全可观的。

11.2 解耦控制

在多变量系统中，不同的输入和输出之间存在着耦合，即系统的第一个输入量不但会对第一个输出量产生影响，而且还会影响到其他的输出量。这样就造成了控制系统设计和实际操作的困难。因此，控制领域的工程人员就提出了解耦的思想，试图把多变量系统分解为多个单变量系统。

解耦控制又称为互不影响控制、一对一控制。最早的工作是由 E.G.Eilbert 完成的。当时称为 Morgan 问题。解耦问题是多输入量多输出量（MIMO）线性定常系统综合理论的一个重要组成部分。其目的是寻找合适的控制规律使闭环控制系统实现一个输出分量仅仅受一个输入分量控制，而且不同的输出分量受不同的输入分量控制。

假设多变量系统的模型描述为（A，B，C，D），其中 A 矩阵是 n×n 维，B 矩阵是 n×p 维，C 矩阵是 m×n 维，D 矩阵是 m×p 维。在解耦系统中还要求 m=p，这是为了保证一个输入量对应一个输出量。则相应的解耦闭环传递函数为：

$$G(s)=(C-DF)(sI-A+BF)^{-1}BR+DR$$

其中 R 是 m×m 维输入变换矩阵，F 是 m×n 维状态反馈矩阵。

如果求得的变换传递函数 G（s）是对角的非奇异矩阵，则该系统是动态解耦的系统；若 G（0）是对角的非奇异矩阵，且系统是稳定的，则称该系统为静态解耦的系统。R 和 F 矩阵的求解可以参看相关的控制理论书籍，这里不加证明的给出其表达式：

$$\begin{cases} R=D_0^{-1} \\ F=D_0^{-1}L \end{cases}$$

$$D_0=\begin{pmatrix} d_1^T \\ \vdots \\ d_m^T \end{pmatrix}=\begin{pmatrix} c_1^T A^{a_1-1}B \\ \vdots \\ c_m^T A^{a_m-1}B \end{pmatrix} \qquad L=\begin{pmatrix} c_1^T A^{a_1} \\ \vdots \\ c_m^T A^{a_m} \end{pmatrix}$$

其中 a_i 称作系统的解耦阶常数，其定义为

$$a_i=\begin{cases}\min[k\,|\,c_i^T A^{k-1}B\neq 0,\ (1\leqslant k\leqslant n)]\\ n\ \ (当\ c_i^T A^r B=0, r=0,1,2\text{L}\ ,n-1)\end{cases}$$，c_i^T 是 C 矩阵的第 i 个行向量。

解耦后的系统还可以进行 a 阶极点配置，具体的公式推导和证明这里就不讨论了。关于解耦控制的 MATLAB 实现请看下例：

某多输入多输出系统状态方程如下。其控制要求为：

(1) 试求解其传递函数矩阵，并讨论系统的运动情况。

(2) 设计解耦控制器，有可能的话进行极点配置。

结果：原系统的传递函数矩阵为

$$A=\begin{pmatrix}0&0&0\\0&0&1\\-1&-1&-3\end{pmatrix}\qquad B=\begin{pmatrix}1&0\\0&0\\0&1\end{pmatrix}$$

$$C=\begin{pmatrix}1&1&0\\0&0&1\end{pmatrix}\qquad D=\begin{pmatrix}0&0\\0&0\end{pmatrix}$$

```
num1 =   0      1.0000     3.0000    -0.0000
         0        0       -1.0000    -0.0000
den1 =   1        3          1          0
num2 =   0        0        1.0000       0
         0      1.0000    -0.0000       0
den2 =   1        3          1          0
```

即
$$G(s)=\frac{1}{s^3+3s^2+s}\begin{pmatrix}s^2+3s & s\\ -s & s^2\end{pmatrix}$$

解耦后系统的传递函数矩阵为

```
numd1 =  0      1       0       0
         0      0       0       0
dend1 =  1      0       0       0
numd2 =  0    0.0000  -0.0000  -0.0000
         0    1.0000   0         0
dend2 =  1      0      0         0
```

即
$$G(s)=\frac{1}{s^3}\begin{pmatrix}s^2&0\\0&s^2\end{pmatrix}=\frac{1}{s}\begin{pmatrix}1&0\\0&1\end{pmatrix}$$

相应的输入变换矩阵 R 和状态反馈矩阵 F1 为

```
R =   1     0
      0     1
F1 =   0     0     1
      -1    -1    -3
```

分析：对于多输入多输出系统，一般都要进行解耦控制或着对角化处理。对于本例，系统的状态方程模型已知，采用上边分析的反馈解耦的方法比较好。并且所有相关的运算都是矩阵运算，特别适用于 MATLAB 求解。不过这种反馈解耦的方法求得的解耦系统一般都是不

同阶次的积分器类型的，真正要投入使用，必须要对相关的闭环极点位置进行配置，以获得期望的性能指标。

求解过程：

本例的解题步骤分为以下几步：

1．求解原系统的传递函数及输入对输出的阶跃响应

首先要根据系统的传递函数判断是否解耦，然后根据其不同输入与输出之间的阶跃响应曲线进行验证。在 MATLAB Editor/Debugger 下输入以下代码：

```
A=[ 0 0 0;
     0 0 1;
    -1 -1 -3];
B=[1 0;
   0 0;
   0 1];
C=[1 1 0;
   0 0 1];
D=zeros(2,2);
%求解传递函数矩阵
[num2,den2]=ss2tf(A,B,C,D,2);
[num1,den1]=ss2tf(A,B,C,D,1);
%绘制不同输入量的阶跃响应曲线
t=0:0.1:10;
u=ones(size(t));
step(A,B,C,D,1,t);
step(A,B,C,D,2,t);
```

相应求得的系统传递函数矩阵请参看上一部分“结果”中的数据。

从“结果”的传递函数矩阵可以看出，系统的耦合比较严重。两个输入量对两个输出量的传递函数均不为零，因此相应的响应曲线也不为零，相互之间存在影响。图 11-12 是系统对第一个输入量的阶跃响应曲线，其上半部分是第一个输出量的响应曲线，下半部分是第二个输出量的响应曲线。图 11-13 是系统对第二个输入量的阶跃响应曲线，其结构与图 11-12 相同。

从这两张图中也可以看出，输出量与输入量之间存在耦合，任一个输入量对两个输出量均有影响。如果不设计解耦控制器，这样的系统是很难控制的，其计算量随阶次的升高以指数形式增加。

2．解耦控制器设计

解耦控制器设计的关键是求解解耦阶常数 a_i。有了 a_i 之后，输入变换矩阵 R 和状态反馈矩阵 F 就非常容易求解了。当然，严格说来，在求解解耦控制器之前还应该判断系统的可控性和可观性，如果系统不完全可控可观的话是无法进行解耦控制器的设计的。为了节省空间起见这里就略去了，因为判断系统可控性和可观性的 MATLAB 程序在前文已经有较详细的叙述了。紧接上一步，输入以下代码：

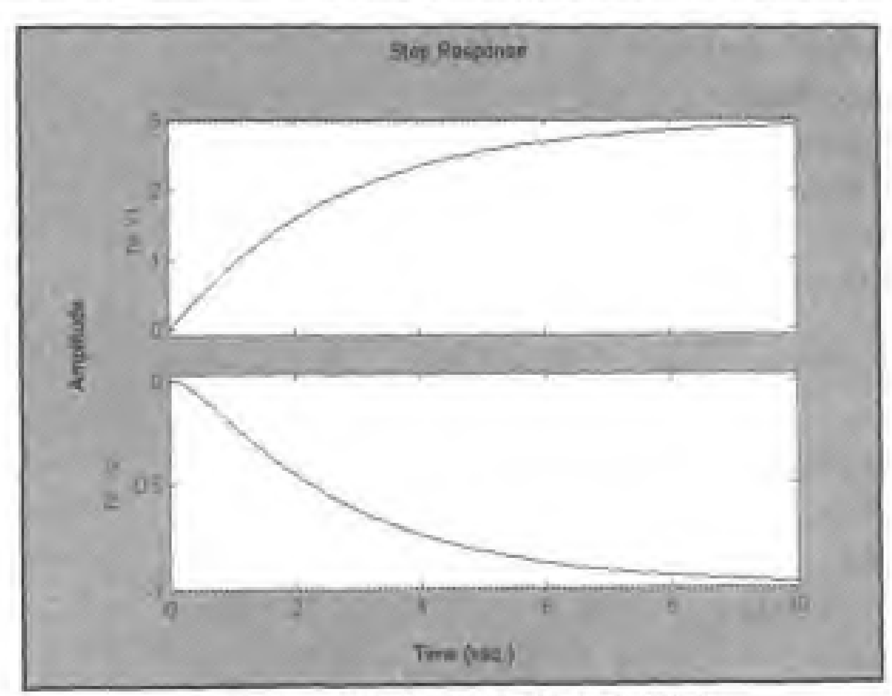


图 11-12　系统对第一输入量的阶跃响应曲线

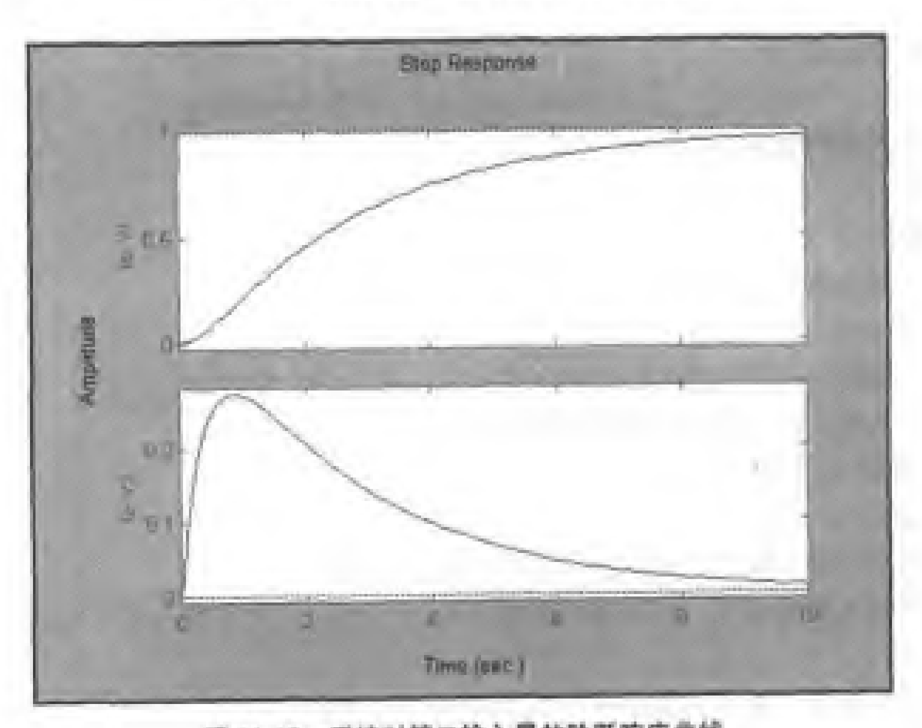


图 11-13　系统对第二输入量的阶跃响应曲线

```
[m,n]=size(C);
D0=C(1,:)*B;
%解耦阶常数 a
a(1)=0;
%求解解耦阶常数 a、矩阵 D0
for i=2:m
    for j=0:n-1
```

```
            D0=[D0;C(i,:)*A^j*B];
            if rank(D0)==i,
                a(i)=j;
                break;
            else,
                D0=D0(1:i-1,:);
            end
        end
    end
    %输入变换矩阵 R
    R=inv(D0);
    %求解矩阵 L
    L=C(1,:)*A^(a(1)+1);
    for i=2:m
        L=[L;C(i,:)*A^(a(i)+1)];
    end
    %求解状态反馈矩阵
    F1=R*L;
    %解耦后系统的状态方程
    B1=B*R;
    A1=A-B*F1;
    %系统解耦后的传递函数矩阵
    [numd1,dend1]=ss2tf(A1,B1,C,D,1);
    [numd2,dend2]=ss2tf(A1,B1,C,D,2);
    %解耦后系统对两个输入量的阶跃响应
    step(A1,B1,C,D,1,t);
    step(A1,B1,C,D,1,t);
```

解耦后系统对第一输入量的阶跃响应曲线如图 11-14 所示。

此时第一输入量仅对第一输出量有影响，对第二输出量的影响为零，实现了对第一输入量的解耦。

解耦后系统对第二输入量的阶跃响应曲线如图 11-15 所示。

同样，此时第二输入量仅对第二输出量有影响，对第一输出量的影响为零，实现了对第二输入量的解耦。

相应求得的解耦后系统传递函数矩阵和状态反馈矩阵 F1 请参看上一部分“结果”中的数据。该传递函数矩阵是非奇异的对角矩阵，因此系统已经实现完全解耦。第一个输入量仅控制第一个输出量，第二个输入量仅控制第二个输出量。不过还应该看到，该矩阵的对角元素都是一阶积分器，其作用是对输入量作无穷积分。因此，如果没有限制的话，该系统是不稳定的。并且，由于两个传递函数相同，系统对第一个输入量的响应和第二个输入量的响应完全相同。

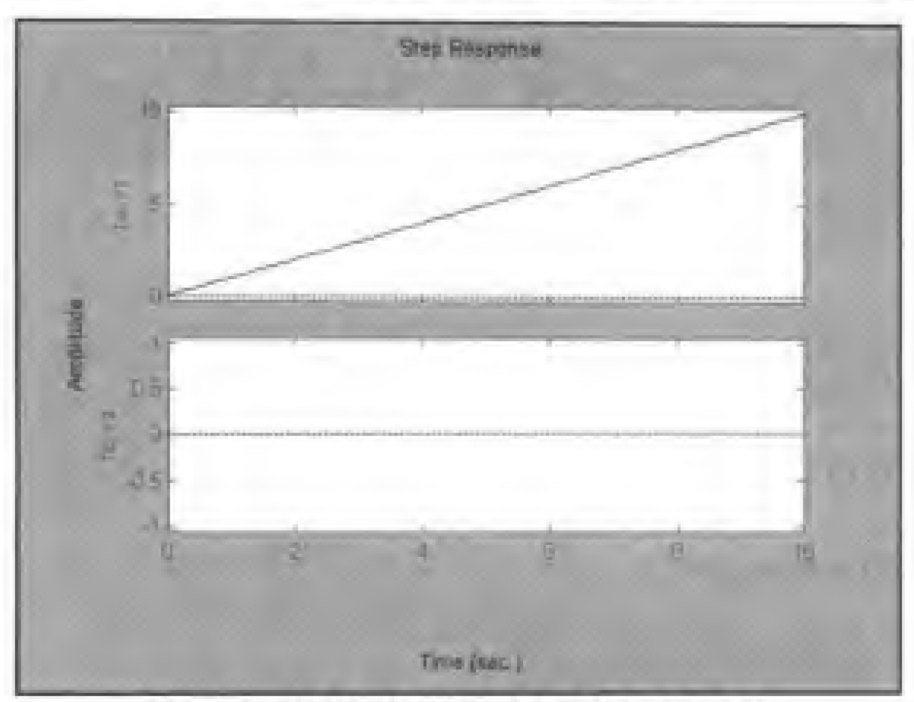


图 11-14　解耦后系统对第一输入量的阶跃响应曲线

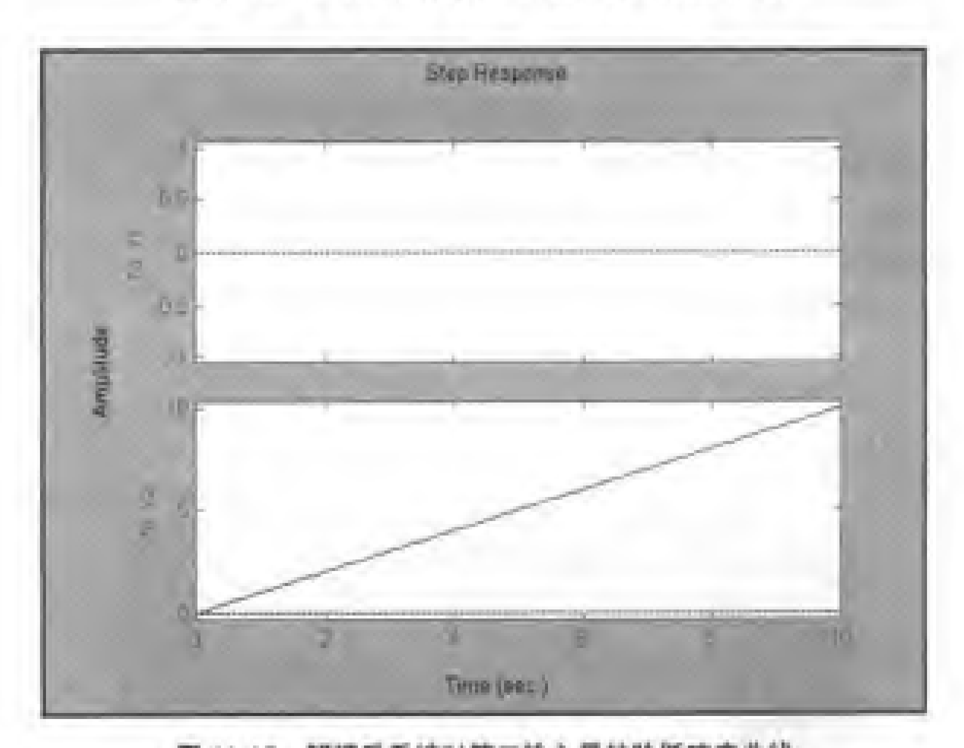


图 11-15　解耦后系统对第二输入量的阶跃响应曲线

从图 11-14 和图 11-15 的阶跃响应曲线上我们也验证了这种说法。一方面，第二个输出量对第一个输入量的阶跃响应曲线为零，第一个输出量对第二个输入量的阶跃响应曲线也为零；另一方面，第一和第二个输出量对各自输入量的响应曲线是一条斜率为 1 的直线，这正是一阶积分器的阶跃响应曲线。除了特殊情况下，一般这样的相同是不能投入使用的，应该进行解耦系统的极点配置。

3. 解耦系统 a 阶极点配置

多输入多输出系统解耦后形成了 m 个单输入单输出的子系统。如果第 i 个子系统的传递

函数是 a_i 阶积分器，那么就可以通过状态反馈矩阵配置 a_i 个闭环极点，称为 a 阶极点配置。状态反馈矩阵的求法与 4.1.1 节介绍的相同。本例都是一阶积分器，所以这两个子系统各可以配置一个极点。我们将第一个子系统的极点配置在-2，第二个子系统极点配置在-3。紧接上一步，输入以下代码：

```
%期望的闭环极点
beta1=-2;
beta2=-3;
beta=[beta1,beta2];
%求解变换矩阵 L
L=zeros(size(C));
for i=1:m
    L(i,:)=C(i,:)*A-beta(i)*C(i,:);
end
%求解极点配置的状态反馈矩阵 F2
F2=R*L;
%求解配置后的系统状态方程
A2=A-B*F2;
B2=B*R;
%配置极点后系统的传递函数矩阵
[numdp1,dendp1]=ss2tf(A2,B2,C,D,1);
[numdp2,dendp2]=ss2tf(A2,B2,C,D,2);
%配置极点后系统的阶跃响应曲线
t=0:0.01:2;
step(A2,B2,C,D,1,t);
step(A2,B2,C,D,2,t);
```

图 11-16 是进行 a 阶极点配置后系统对第一个输入量的阶跃响应曲线。可以看出，在系

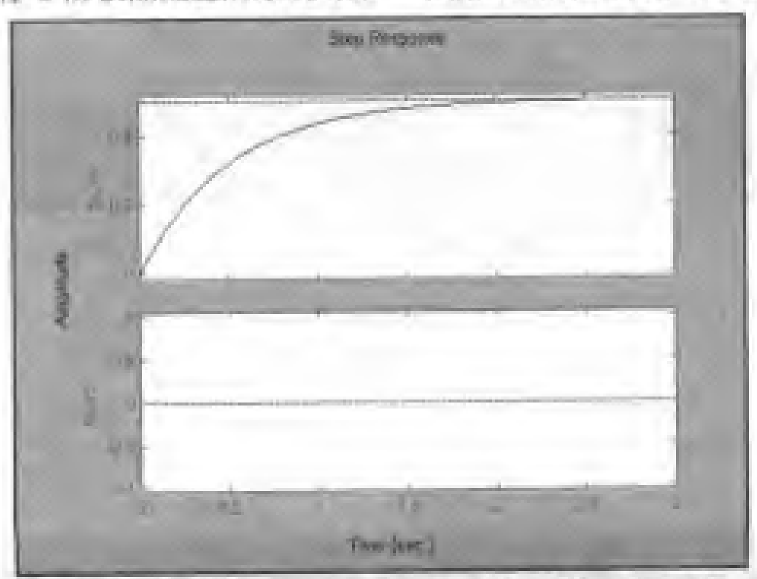


图 11-16 配置极点后系统对第一个输入量的阶跃响应曲线

统输出解耦的情况下实现了一阶极点配置，响应曲线满足要求。此时系统对第一个输入量的阶跃响应曲线见图 11-17。

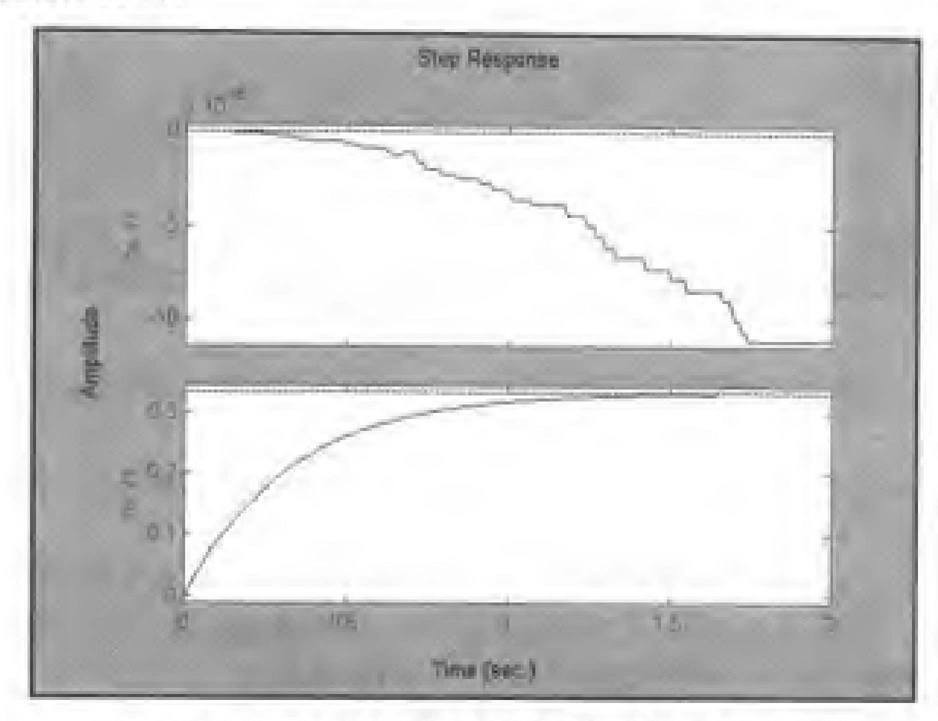


图 11-17　配置极点后系统对第二个输入量的阶跃响应曲线

可以求得系统此时的传递函数矩阵为

```
numdp1 =    0      1        3        0
            0      0        0        0
dendp1 =    1      5        6        0
numdp2 =    0   -0.0000  -0.0000  -0.0000
            0    1.0000   2.0000     0
dendp2 =    1      5        6        0
```

即

$$G(s)=\frac{1}{s^3+5s^2+6s}\begin{pmatrix} s^2+3s & 0 \\ 0 & s^2+2s \end{pmatrix}=\begin{pmatrix} \frac{1}{s+2} & 0 \\ 0 & \frac{1}{s+3} \end{pmatrix}$$

可见，系统不但完全解耦，而且分别将两个解耦后的子系统的闭环极点配置在了-2 和-3。并且，图 11-16 和图 11-17 的响应曲线也符合要求，一方面实现了解耦控制，另一方面各子系统是稳定的。

有的读者可能会问，图 11-17 的第一个输出量对第二个输入量的响应曲线并不为零，是不是没有实现解耦控制？其实不是这样的，仔细看一下图 11-17 上半部分的纵坐标就会发现，这条曲线纵坐标的数量级是 10 的-16 次方。可以肯定地说，这是由于 MATLAB 数值求解的误差造成的。当然，这种误差在工程领域是可以完全忽略不计的。并且，即使是这样小的误差，最终也会限制在很小的范围内。我们可以将横轴的时间坐标放大到 1000s，重新绘制这根曲线，如图 11-18 所示。

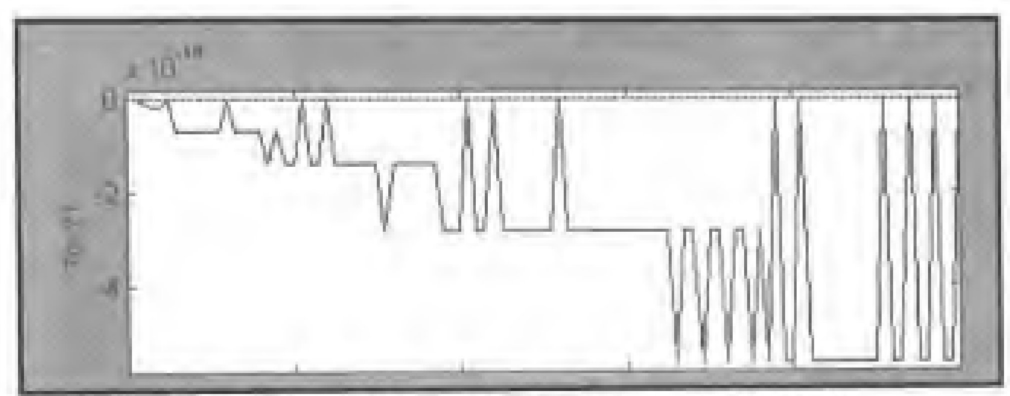


图 11-18　第一个输出量对第二个输入量的阶跃响应曲线

可见，响应误差最终稳定在 10 的－14 次方数量级上，对最终解耦控制器的正常运行不会产生影响。

小结：本例比较详细地分析了多输入多输出系统的解耦控制器设计和极点配置的问题，并且给出了响应的 MATLAB 程序。

事实上，多变量系统的分析和求解还有许多其他的方法，例如多变量频域控制领域的对角优势法、逆 Nyquist 图法等等。因为都是依据矩阵运算和图形处理，所以 MATLAB 同样可以编写出这些算法的程序。本节并没有介绍新的 MATLAB 函数，从总体来看，到目前为止除了有关最优化的函数外，我们介绍的 MATLAB 函数已经足以胜任有关控制理论基础课程的学习和实践需要了。关键是掌握算法和思路，应用 MATLAB 这个得力的工具解决实际问题。

11.3　线性二次型最优控制器设计

对于线性系统，若取状态变量和控制变量的二次型函数的积分作为性能指标，这种动态系统最优化问题称为线性系统二次型性能指标的最优控制问题，简称线性二次型问题。它的最优解可以写成统一的解析表达式，而且可以导出一个简单的状态线性反馈控制律，其计算和工程实现都比较容易。

MATLAB 控制系统工具箱中提供了一些 LQ（Linear Quadratic，线性二次型）设计工具，可以很方便地完成线性二次型最优控制器的设计。下面我们简要介绍一下有关的理论内容，然后给出一个应用实例。

11.3.1　代数 Riccati 方程求解

设线性系统的状态方程模型（A，B，C，D）已知，如果希望这样一个系统能够满足某种最优的要求，最简单的可以引入线性二次型最优控制指标，即

$$J=\frac{1}{2}X^{T}(tf)SX(tf)+\frac{1}{2}\int_{t0}^{tf}\left[X^{T}(t)Q(t)X(t)+u^{T}(t)R(t)u(t)\right]\mathrm{d}t$$

其中 Q（t）和 R（t）分别是对状态变量和控制量的加权矩阵。一般情况下，假定这两个矩阵为定常矩阵（即不随时间变化），并简记 Q（t）$=Q$，R（t）$=R$。线性二次型最优控制就是求出 J 最小时的控制量 u（t），从而获得性能最优。为了达到这一目的，首先构造一个 Hamilton 函数：

$$H = -\frac{1}{2}\left[X^T(t)QX(t) + u^T(t)Ru(t)\right] + \lambda^T\left[AX(t) + Bu(t)\right]$$

然后通过求导的方法可以求出最优控制信号 u（t）为：

$$u(t) = -R^{-1}B^T P(t)X(t)$$

其中 P（t）矩阵就是 Riccati 方程的解：

$$P(t) = -P(t)A - A^T P(t) + P(t)BR^{-1}B^T P(t) - Q$$

上面的方程是微分 Riccati 方程，一般是多个相互耦合的非线性微分方程组，除了特殊情况外，一般不存在解析解。这就给求解最优控制信号 u（t）造成了困难。因此，我们一般求解 u（t）的稳态解。即令 tf 趋于无穷，则 P（t）趋于一个常值矩阵，P（t）的一阶导数趋于零，有：

$$0 = -PA - A^T P + PBR^{-1}B^T P - Q$$

上式被称为代数 Riccati 方程,其求解就比较容易了。并且,因为都是矩阵运算,用 MATLAB 实现起来也比较容易。并且，MATLAB 提供了一条求解代数 Riccati 方程的函数 care（），省去了我们手工编程的工作。其基本调用格式为

[X,L,G,RR] = CARE(A,B,Q,R,S,E)

它所求解的代数 Riccati 方程形式为

$$A^T XE + E^T XA - (E^T XB + S)R^{-1}(B^T XE + S^T) - Q = 0$$

一般缺省设置 S=0，E=I，这样该方程就和原始的代数 Riccati 方程一致了。X 是求得的代数 Riccati 方程的解，L 是闭环状态方程参数矩阵的特征值，G 是状态反馈矩阵，RR 是残留矩阵的 Frobenius 范数。有了代数 Riccati 方程的解，我们就可以求出最优控制信号 u（t）和系统的响应曲线了。请看下例：

某控制系统的状态方程描述以及 Q、R 矩阵如下。试求解其代数 Riccati 方程的解和最优控制信号 u（t），并绘制系统对两个输入量的阶跃响应曲线。

$$A = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & -5 & 1 & 0 \\ 0 & 0 & -5 & 1 \\ 0 & 0 & 0 & -5 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 0 \\ 0 & 5 \\ 0 & 3 \\ 1 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & & & \\ & 2 & & \\ & & 3 & \\ & & & 4 \end{pmatrix} \quad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

结果：系统的代数 Riccati 方程的解为

```
X =   0.2099    0.0018    0.0006   -0.0192
      0.0018    0.1549   -0.0270   -0.0215
      0.0006   -0.0270    0.2545    0.0048
     -0.0192   -0.0215    0.0048    0.3809
```

分析：本例是一个 2×2 的多输入多输出系统，因此最优控制信号是二维的。Q 是状态变

量的加权矩阵，R 是控制信号的加权矩阵，其意义是衡量二者在性能指标 J 里的权重。除了要求它们必须是正定矩阵之外没有什么特殊要求。不过不同 Q 和 R 会得出不同的代数 Riccati 方程的解，进而会得出不同的最优控制信号 $u(t)$。下一节我们将具体讨论 Q 和 R 的选取规则。

求解过程：

本例的解题步骤分为以下几步：

1．求解代数 Riccati 方程

调用 care（）函数可以很快得到代数 Riccati 方程的解。为了比较详细的阐述线性二次型最优控制器的求解步骤，我们调用简单格式的 care（）函数。在 MATLAB Editor/Debugger 下输入以下代码：

```
%清除所有内存变量
clear all;
%系统的状态方程
A=[-2  0   0   0;
     0 -5   1   0;
     0  0  -5   1;
     0  0  0   -5];
B=[2 0;
   0 5;
   0 3;
   1 1];
C=[1 1 0 0;
   0 1 2 1];
D=zeros(2);
%Riccati 方程的 Q、R 矩阵
Q=diag(1:4);
R=eye(2);
%求解代数 Riccati 方程的解
X=care(A,B,Q,R);
```

求得的本系统的代数 Riccati 方程的解请参看上一部分“结果”中的数据。

2．求解系统的最优控制信号

根据最优控制信号的表达式：

$$u(t)=-R^{-1}B^{T}PX(t)$$

得到系统的代数 Riccati 方程的解之后可以求出最优控制信号的变化曲线，其中，状态反馈矩阵 K 等于：

$$K=-R^{-1}B^{T}P$$

紧接上一步，输入以下代码：

```
%系统的状态反馈矩阵
K=-inv(R)*B'*X;
%仿真时间
```




```
t=0:0.005:0.5;
%系统的状态方程抽象描述
sys=ss(A+B*K,B,C,D);
%求解系统的状态变量 x(t)
[y,T,xt]=step(sys);
%求解最优控制信号 u(t)
n=length(T);
m=length(R);
for i=1:m
    for j=1:n
        u(j,:,i)=K(i,:)*(xt(j,:,i))';
    end
end
%图形绘制
subplot(211),
%最优控制信号 u(1)
plot(T,u(:,:,1))
title('Linear Quadratic Controller output u1(t)')
xlabel('Time-sec')
ylabel('Value');
grid;
subplot(212),
%最优控制信号 u(2)
plot(T,u(:,:,2))
title('Linear Quadratic Controller output u2(t)')
xlabel('Time-sec')
ylabel('Value');
grid;
```

得到系统的二维最优控制信号如图 11-19 所示。

从图 11-19 中可以看出，在阶跃输入下，控制信号基本上在 0.5s~0.7s 左右就趋于稳定，也就是说，系统的过渡过程时间不会大于这个值。当然，在实际的控制系统中，最优信号的值是不必我们手工计算的，只需求出状态反馈矩阵，也就是最优控制器即可。这里求解出系统的最优控制信号变化曲线，一方面是帮助读者理清最优控制器的设计思路，另一方面也是系统运动状态的一种反映。

3．系统对输入的响应曲线

紧接上一步，在 MATLAB Command Window 键入下列语句即可获得图 11-20 和图 11-21 所示的曲线：

```
step(A+B*K,B,C,D,1,t)
step(A+B*K,B,C,D,1,t)
```

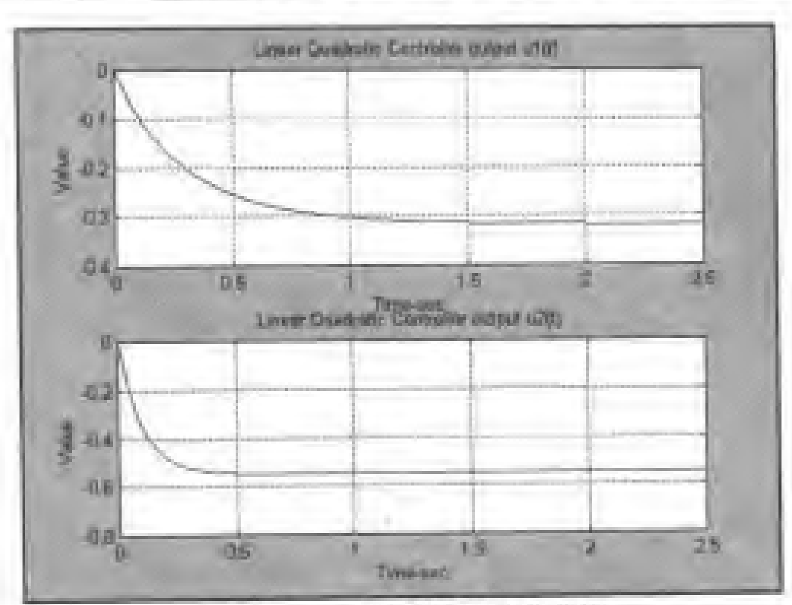


图 11-19　最优控制信号变化曲线

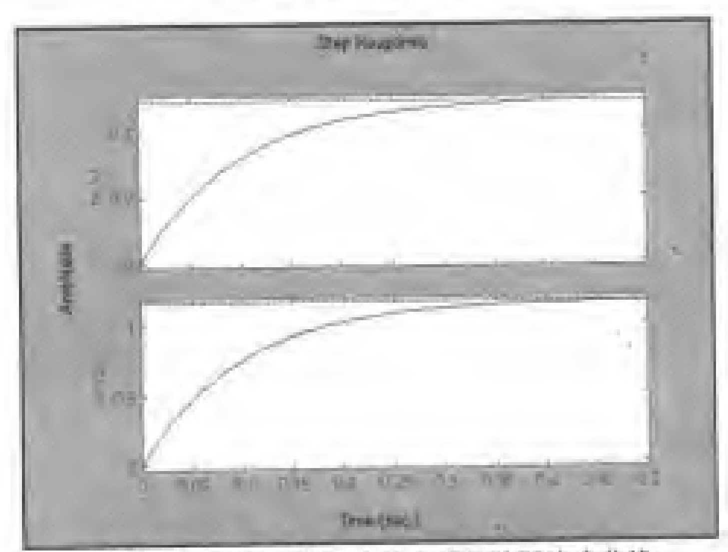


图 11-20　系统对第一个输入量的阶跃响应曲线

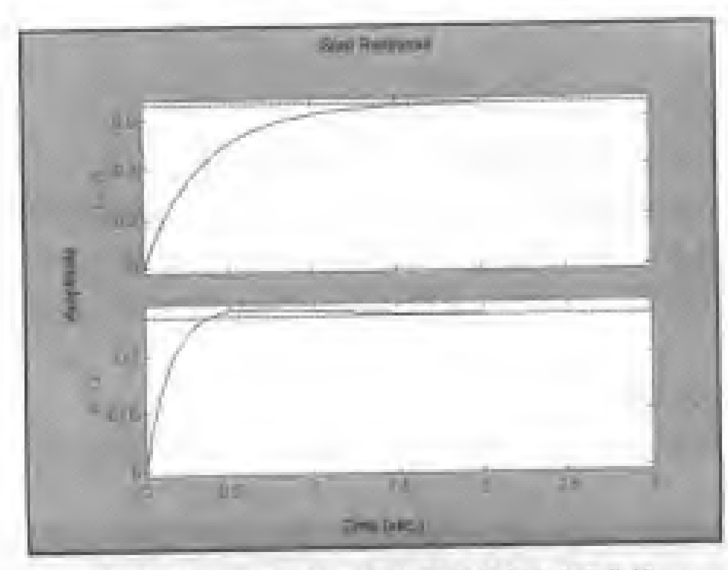


图 11-21　系统对第二个输入量的阶跃响应曲线

这里求解系统的闭环参数矩阵时用的是 A+B*K 而不是 A-B*K，这是因为在计算状态反馈矩阵 K 时已经包含负号了。从图 11-20 和图 11-21 中还可以看出，系统的过渡过程时间都在 0.5s~0.7s 之间，这也验证了第二步的说法。

小结：本例的重点是使用 MATLAB 求解代数 Riccati 方程的解及其在求解最优控制信号中的应用，因此没有对系统进行解耦控制和性能指标的设计。有关这方面的内容，我们将在下一小节讨论。

11.3.2　线性二次型最优控制器设计举例

使用线性二次型最优控制器进行控制系统设计和校正的最大优点就是不必根据要求的性能指标确定闭环极点的位置，只需根据系统的响应曲线寻找出合适的状态变量和控制量的加权矩阵即可。因为求得的控制器是误差指标 J 最优意义下的控制器，所以系统的性能也是 J 指标意义下最优的。

某倒立单摆系统如图 11-22 所示，其状态方程已知。

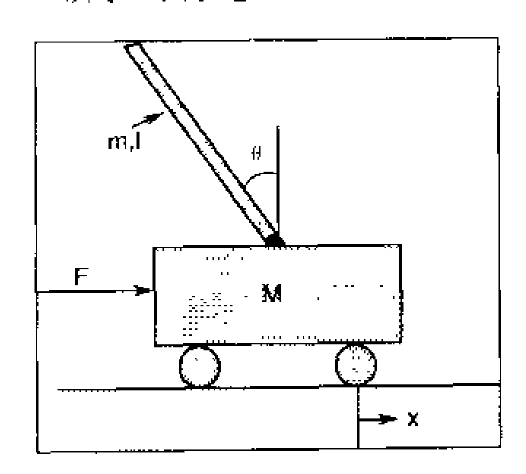


图 11-22　倒立单摆系统示意图

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \dfrac{-\left(I+ml^2\right)b}{I(M+m)+Mml^2} & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 0\\ 0 & \dfrac{-mlb}{I(M+m)+Mml^2} & \dfrac{mg\,l(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix}=\begin{bmatrix}0\\ \dfrac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \dfrac{-ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}u$$

其中小车的质量为 M=0.5kg，倒立单摆的质量为 m=0.5kg，小车的摩擦系数为 b=0.1，端点与倒立单摆质心的距离为 l=0.3m，倒立单摆的惯量为 I=0.006 kg · m^2，输入量 u=F 是施加在小车上的外力，四个状态变量分别是小车的坐标 x，x 的一阶导数，倒立单摆的垂直角

度 Φ，Φ的一阶导数。输出的被控量分别是小车的坐标 x 和倒立单摆的垂直角度 Φ。试根据误差指标 J 最优意义下最优的规则设计线性二次型最优控制器和相关的参考输入以及观测器，满足以下指标：

(1) 输出量 x 和 Φ的过渡过程时间小于 2s。

(2) 输出量 x 的上升时间小于 0.5s。

(3) 输出量 Φ的超调量小于 20°（0.35 rad/s)。

结果：求得的线性二次型最优控制器为：

K = -70.7107 -37.8345 105.5298 20.9238

分析：本例是一个单输入双输出系统．采用属于经典控制方法的 PID 校正和根轨迹校正都无法满足同时控制两个输出量的要求。如果采用状态空间的极点配置方法，必须分别设计两个反馈校正装置，根据不同的性能指标分别配置极点。而采用 LQ 方法就不必这样繁琐，只需设计一个状态反馈校正装置，在性能指标 J 最小的意义下求得最优控制信号即可。MATLAB 还提供了一条直接求解 LQ 最优状态反馈的函数 lqr()，我们可以不必根据 care() 函数求解的数据进行计算。其基本调用格式为：

[K,S,E] = LQR(A,B,Q,R)

其中 A，B，Q，R 的意义如前所述，K 是求得的最优状态反馈矩阵，S 是相应的代数 Riccati 方程的解，E 是闭环系统的特征值。

求解过程：

本例的解题步骤分为以下几步：

1．分析原系统的开环阶跃响应

首先要求得到其开环系统的特征值，判断其稳定性。然后根据其阶跃响应曲线分析当前的运动情况与期望性能指标之间的差距，确定校正的手段。在 MATLAB Editor/Debugger 下输入以下代码：

```
%清除所有内存变量
clear all;
%系统参数初始化
M = 0.5;
      m = 0.2;
      b = 0.1;
      i = 0.006;
      g = 9.8;
      l = 0.3;
%系统的状态方程描述
      p = i*(M+m)+M*m*l^2;
      A = [0        1                  0                 0;
           0   -(i+m*l^2)*b/p   (m^2*g*l^2)/p          0;
           0        0                  0                 1;
           0   -(m*l*b)/p         m*g*l*(M+m)/p          0];
      B = [0; (i+m*l^2)/p; 0; m*l/p];
```




```
    C = [1 0 0 0;
         0 0 1 0];
    D = [0;0];
   %求解系统的特征值
    p = eig(A);
    t=0:0.01:1;
    step(A,B,C,D,1,t)
```

求得系统的特征值为：

```
p =      0
    -0.1428
     5.5651
    -5.6041
```

系统有一个右半平面的极点，因此不稳定，必须加入校正装置。此时系统的阶跃响应曲线如图 11-23 所示。

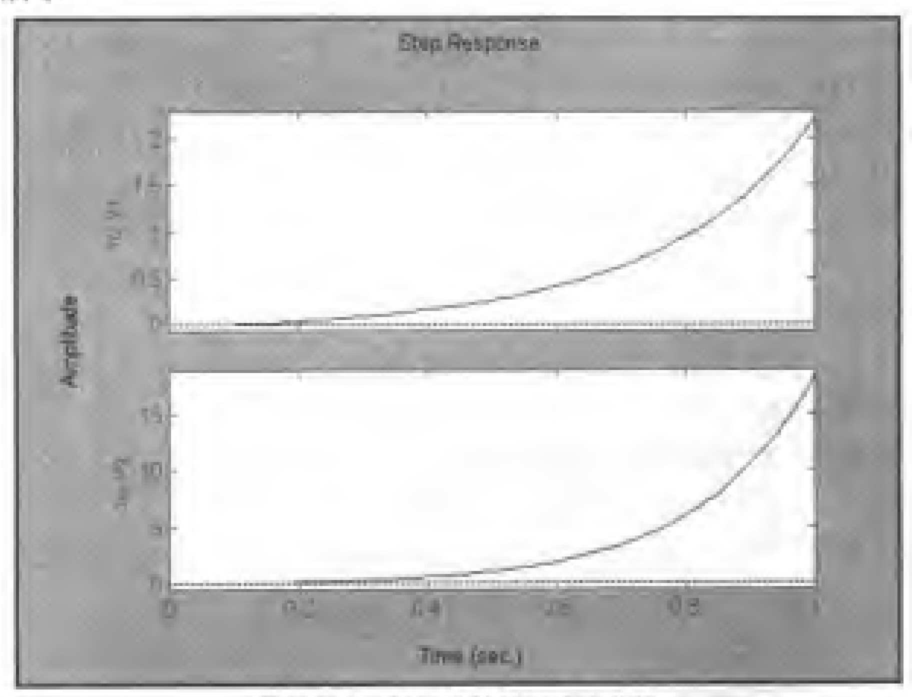


图 11-23 原系统的阶跃响应曲线

图 11-23 上半部分是小车坐标 x 的阶跃响应曲线，下半部分是倒立单摆的垂直角度φ的阶跃响应曲线。从图中可以看出它们都不稳定，谈不上与期望的性能指标作比较。下面加入 LQ 校正装置。

2．线性二次型最优控制器设计

设计线性二次型最优控制器关键是选择加权矩阵 Q。一般说来，Q 选择的越大，系统达到稳态所需的时间越短，当然，还要实际的系统允许。我们首先选择 Q=C'*C，R=I，然后根据实际情况进行调节。输入以下代码：

```
%Q 和 R 矩阵的选择
x=1;
```



```
y=1;
Q=[x 0 0 0;
   0 0 0 0;
   0 0 y 0;
   0 0 0 0];
R = 1;
%求解线性二次型最优状态反馈矩阵
K = lqr(A,B,Q,R)
%求解系统闭环状态方程
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
%输出系统阶跃仿真
T=0:0.02:2;
U=ones(size(T));
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,T);
plot(T,Y(:,1),'.-',T,Y(:,2));
title('Inverted Pendulum LQ Step Response');
xlabel('Time-sec');
Ylabel('Response');
grid;
legend('Cart','Pendulum');
```

此时求得的线性二次型最优状态反馈矩阵为：

K = −1.0000   −1.6567   18.6854   3.4594

从图 11-24 系统的阶跃响应曲线中可以看到，系统超调量基本满足要求，但一方面系统

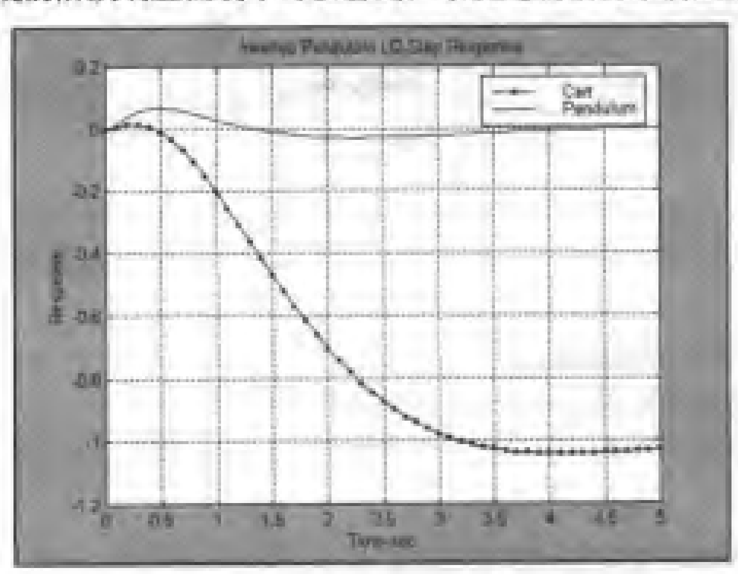


图 11-24　初步 LQ 校正后系统的阶跃响应曲线

的稳态值与期望相差很远（小车坐标的响应曲线稳态值为负值），另一方面过渡过程时间和上升时间都很大，必须重新校正。方法就是加大加权矩阵 Q 的值。经不断仿真发现，当 x=5000，y=100 时效果比较理想。因此，在 MATLAB Editor/Debugger 下将上一段代码中的 x 和 y 改成相应的值，重新运行代码，有：

求得的线性二次型最优状态反馈矩阵为：

K =　−70.7107　−37.8345　105.5298　20.9238

可以看到，该状态反馈矩阵要明显大于上一步设计的反馈矩阵。此时系统的阶跃响应曲线如图 11-25 所示。

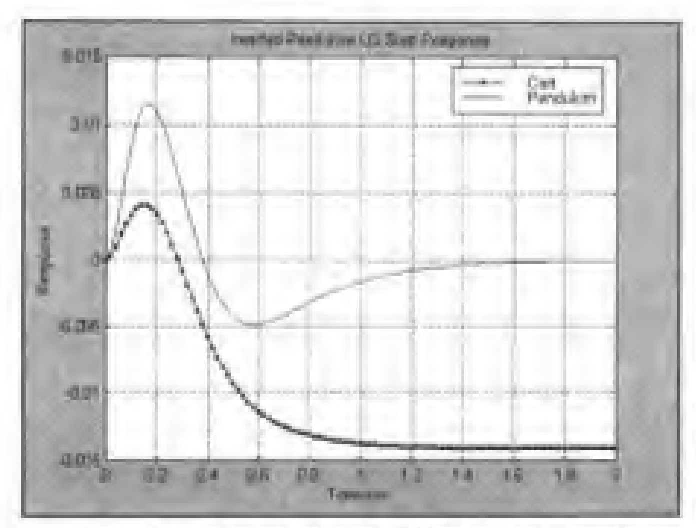


图 11-25　加入 LQ 校正后系统的阶跃响应曲线

从图 11-25 中可以看到，系统响应的快速性得到了明显的改善，上升时间和过渡过程时间都满足最初设计的要求，但稳态值依旧没有得到改善。这是因为没有加入参考输入的原因。因此，为了保持解题的完整性，我们下一步将设计系统的参考输入和状态观测器，使得系统的响应完全符合要求的性能指标。

3. 参考输入及状态观测器设计

参考输入和状态观测器的功能和设计方法在前文已经有详细的解释，这里就不再赘述了。一般说来，经过线性二次型最优化方法设计的控制器投入使用前都要设计参考输入和状态观测器，以满足实际需要。紧接上一步，输入以下代码：

```
%参考输入倍数
Cn=[1 0 0 0];
Nbar=rscale(A,B,Cn,0,K)
Bcn=[Nbar*B];
%观测器极点
P = [-40 -41 -42 -43];
%观测器状态反馈矩阵
L = place(A',C',P)'
%加入观测器和参考输入后系统的状态方程描述
Ace = [A-B*K                  B*K;
          zeros(size(A)) (A-L*C)];
```

```
Bce = [        B*Nbar;
      zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];
%阶跃响应曲线仿真
T = 0:0.02:2;
U = ones(size(T));
[Y,X]=lsim(Ace,Bce,Cce,Dce,U,T);
plot(T,Y(:,1),'.-',T,Y(:,2));
title('Inverted Pendulum LQ Step Response');
xlabel('Time-sec');
Ylabel('Response');
grid;
legend('Cart','Pendulum');
```

对应的观测器状态反馈矩阵为：

```
L =   1.0e+003 *
    0.0826    -0.0010
    1.6992    -0.0402
   -0.0014     0.0832
   -0.0762     1.7604
```

图 11-26 就是最终的系统阶跃响应曲线，两条曲线的过渡过程时间都小于 2s，小车坐标 x 的上升时间小于 0.5s，倒立单摆的垂直角度Φ的第一个正的峰值也小于 0.35rad/s。我们还可以绘制出系统的最优控制信号：

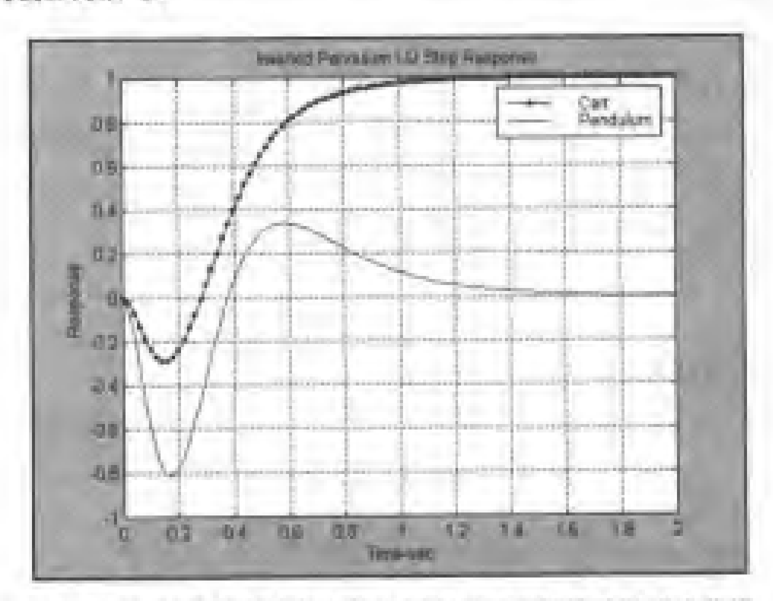


图 11-26　加入参考输入和状态观测器后系统的阶跃响应曲线

相应的代码为：

```
K=[K,0,0,0,0];
```




```
n=length(T);
    for j=1:n
        u(j,:)=K*(X(j,:))';
    end
 plot(T,u);
 title('Inverted Pendulum LQ Controllor output:u(t)');
xlabel('Time-sec');
Ylabel('Response');
grid;
```

从图 11-27 中看到，控制信号最大输出峰值接近-100，实际应用中尤其是在工业控制领域要考虑控制器的阀门的量程限额。不过我们现在仅仅是从理论上阐明系统的运动和校正手段，就不再讨论这个问题了。

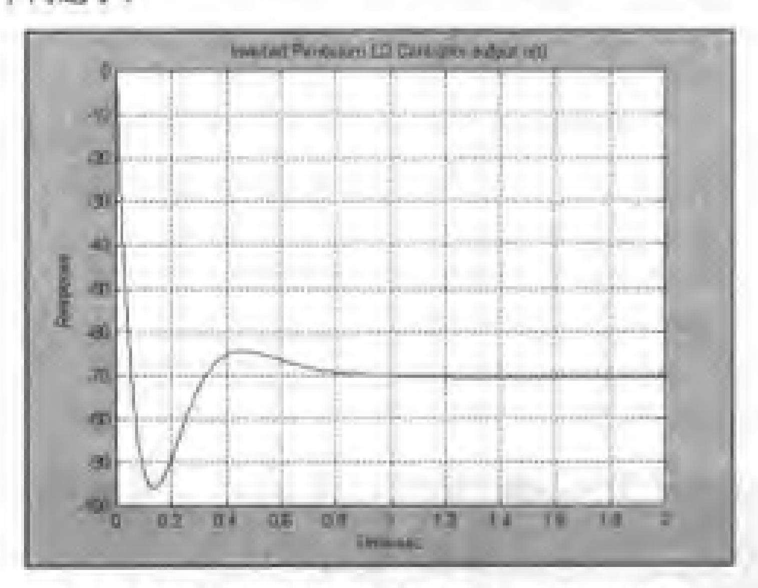


图 11-27 倒立单摆系统的线性二次型最优控制信号

小结：本例详细介绍了有关线性二次型最优控制器的设计方法和 MATLAB 实现手段，重点是 lqr（ ）函数的应用。事实上，使用 care（ ）函数也可以进行线性二次型最优控制器的设计。不过 lqr（ ）函数的缺省返回值是状态反馈矩阵 K，而 care（ ）函数的缺省返回值是代数 Riccati 方程的解 X。本例还使用了一个新的函数 legend（ ），其功能是为 plot（ ）函数绘制的图形增加注释。该函数能够自动识别不同的曲线颜色和绘制的线段类型。其具体参数设置和使用方法请参看 MATLAB 帮助。

本章主要介绍了控制系统各种常用的状态空间设计方法在 MATLAB 下的实现手段，包括通过状态反馈进行极点配置、状态观测器设计、状态空间的解耦控制、代数 Riccati 方程求解、线形二次型最优控制器设计等等。其中最基础也是最常用的是状态反馈进行极点配置和状态观测器设计，其基本思路和实现方法就是控制系统状态空间设计的基本方法。这方面 MATLAB 提供了功能强大的函数，可供我们使用。

第 12 章 神经网络与控制

神经网络是 20 世纪 80 年代末期发展起来的一种实用的多学科交叉技术，已成为“智能控制”的一个新分支，是自动控制领域的前沿学科之一。它为解决复杂的非线性、不确定、不确知系统的控制问题，开辟了一条新的途径。

本章介绍的神经网络控制理论主要是（人工）神经网络理论与控制理论的结合，而神经网络技术发展至今，已经汇集了数学、生物学、神经生理学、脑科学、遗传学、人工智能、计算机科学等多领域学科的理论、技术和方法。

MATLAB 环境下提供的用于神经网络仿真的神经网络工具箱（Neural Networks Toolbox），提供了功能更强大、界面更友好的各种神经网络仿真函数，其主要功能包括：

- 函数逼近与建模（Function approximation and modeling）。
- 信号处理与预测（Signal processing and prediction）。
- 分类与聚类（classification and clustering）。

读者可以在 MATLAB Command Window 下键入“demo”命令，通过图 12-1 的界面了解神经网络工具箱新增的内容。

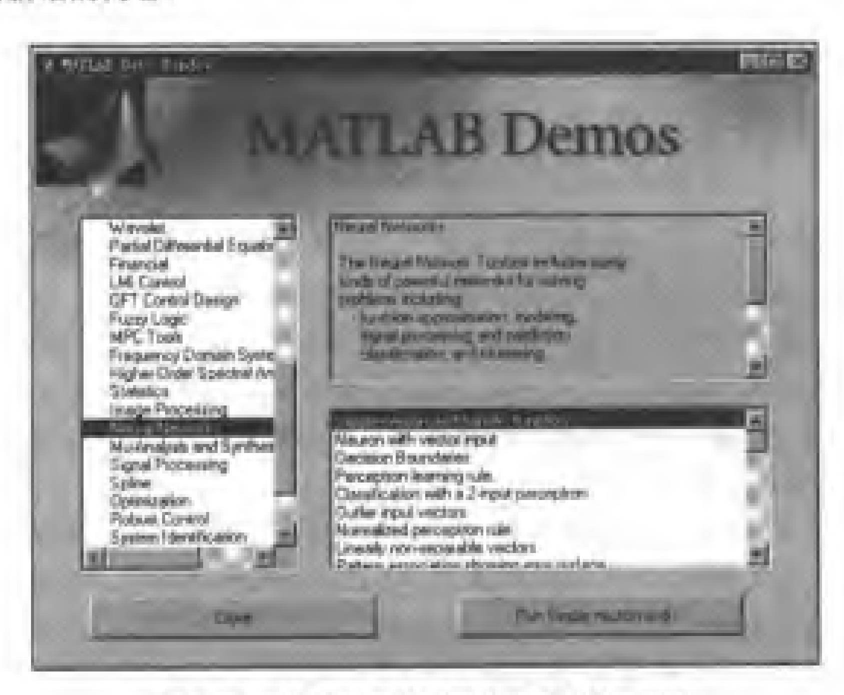


图 12-1 MATLAB 神经网络工具箱演示界面

本章首先介绍一些关于神经网络控制的基础知识，然后结合 MATLAB 环境具体阐述神经网络工具箱的使用方法，最后给出一个应用的实例。

12.1　神经网络概述

人工神经网络（简称神经网络，Neural Networks）是人工神经元（简称神经元，Neuron）相互连接而组成的网络，它是从微观结构和功能上对人脑的抽象、简化，是模拟人类智能的一条重要途径，反映了人脑功能的若干基本特征，如并行信息处理、学习、联想、模式分类、记忆等等。

1943 年建立的第一个神经元模型——MP 模型为神经网络的研究与发展奠定了基础。至今，已经建立了多种神经元与网络的模型，取得了相当多的成果。其中神经网络对控制领域最有吸引力的特性是：

- 能任意逼近 L_2 上的非线性函数；
- 可以实现同步多输入、多输出；
- 便于用超大规模集成电路（VLSI）或光学集成电路系统实现，或用现有的计算机技术实现；
- 能够实现信息的并行分布式处理与存储；
- 能够进行自学习、自调整，适应环境的变化。

图 12-2 是控制领域常用的多层前馈神经网络模型。

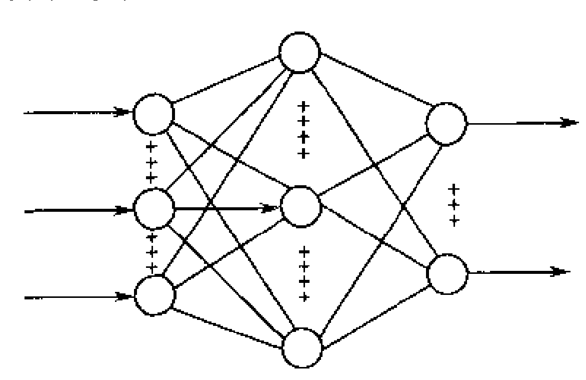


图 12-2　多层前馈神经网络模型

12.1.1　神经网络理论基础

人工神经网络或称连接机制（Connectionism）是源于人脑神经系统的一类模型，它是由简单信息处理单元（人工神经元）相互连接组成的网络。网络的信息处理又通过各处理单元之间的相互作用来实现，具体地说，它是通过把问题表达成处理单元之间的连接权值来处理的。权值大，则该处理单元对后续单元的影响大，反之亦然。

自 20 世纪 80 年代末期以来，学者们建立了多种神经网络模型，其理论框架和思路不外乎下述三大要素：

- 神经元（信息处理单元）的特性；
- 神经元之间的相互见解形式——网络拓扑结构；
- 为适应环境而改善性能的学习规则。

神经网络是具有高度非线性的系统，具有一般非线性系统的特性。虽然单个神经元的组

成和功能极其有限（一般只是权值调整），但大量神经元组成的网络系统，所能实现的功能却是非常丰富多采的。

神经网络的数学模型虽然有很多种，但基本运算可以归结为四类：积与和、权值学习、阈值处理和非线性函数处理。其中关键的部分是权值的学习机制，它是模拟人适应环境的学习过程的一种机器学习模型，大致分为以下三种：

- 有导师学习（SL——Supervised Learning）。在学习过程中，网络根据实际输出与期望输出的比较，进行连接权值的调整。一般将期望输出称为导师信号，它是评价网络学习的标准。
- 无导师学习（NSL——Nonsupervised Learning）。在学习过程中，没有导师信号提供给网络，网络能根据其特有的结构和学习规则，进行连接权值的调整。此时，网络的学习评价标准隐含于其内部。
- 再励学习（RL——Reinforcement Learning）。它把学习看作为试探评价（奖励或惩罚）过程，学习机制选择一个动作（输出）作用于环境后，使环境的状态改变，并产生一个再励信号 r_e（奖励或惩罚）反馈至模型，模型依据再励信号与当前环境的状态，再选择下一个动作作用于环境。动作选择的原则，是使受到奖励的可能性增大。

下面我们以最简单的 MP 模型为例，具体阐述神经网络的工作机制。

MP 模型是 1943 年由美国心理学家 McCulloch 和数学家 Pitts 共同建立的，是最早的人工神经元模型。它是一个多输入——多输出的非线性信息处理单元，其具体的网络拓扑结构如图 12-3 所示。

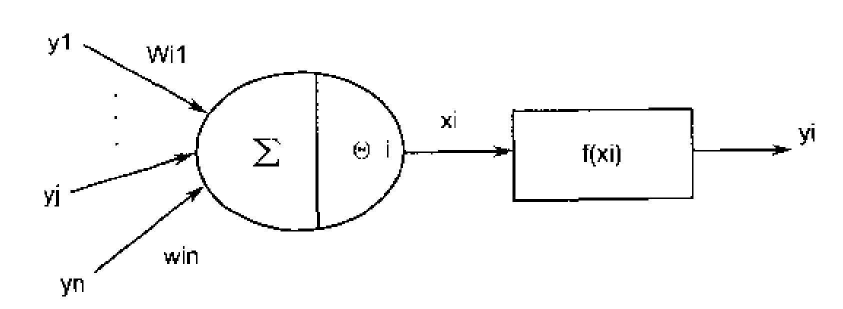


图 12-3　MP 人工神经元模型结构

其中　y_i——神经元 i 的输出，它可以与其他多个神经元通过权相连接；

y_j——与神经元 i 连接的神经元 j 的输出，也是神经元 i 的输入；

w_{ij}——神经元 i 到神经元 j 的连接权值；

θ_i——神经元 i 的阈值；

$f(x_i)$——输出非线性函数。

神经元 i 的输出 y_i 可以用下式描述：

$$y_i = f(\sum_{j=1}^{n} w_{ij} y_j - \theta_i)\ ,\quad i \neq j$$

设

$$x_i = \sum_{j=1}^{n} w_{ij} y_j - \theta_i$$

则有

$$y_i = f(x_i)$$

每一个神经元的输出，或者是数值“0”，或者是数值“1”，分别模拟生物神经的“抑制”或“兴奋”状态。则，输出非线性函数为：

$$f(x)=\begin{cases}1\,, & x\geqslant 0\\ 0\,, & x<0\end{cases}$$

f(x)是一个作用函数（Activation Function），也称激发函数。上式是最简单的阶跃函数形式的作用函数，常用的还有 Sigmoid 型作用函数：

$$f(x)=\frac{1}{1+\mathrm{e}^{-x}}$$

这两种作用函数的图形如图 12-4 所示。

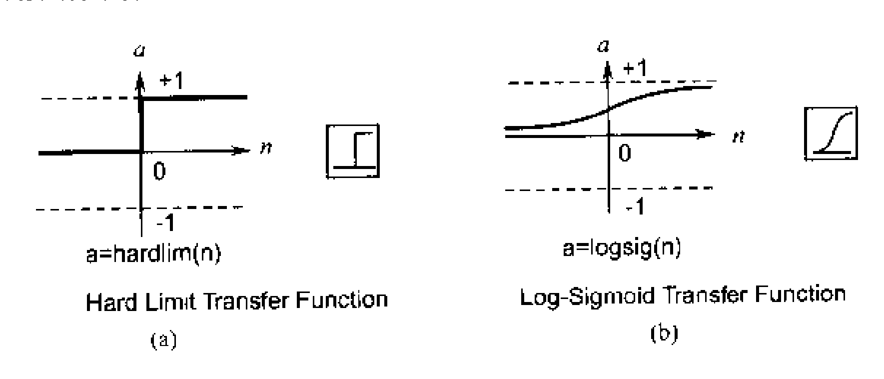


图 12-4　常用的作用函数类型

（a）阶跃函数；（b）Sigmoid 函数。

有时为了满足对称性的要求，也采用下述形式的对称 Sigmoid 型作用函数：

$$f(x)=\frac{1-\mathrm{e}^{-\beta x}}{1+\mathrm{e}^{-\beta x}}\,,\ \beta>0$$

其图像就是将上页图 12-4（b）向下平移 0.5 个坐标单位，并伸长 2 倍，使其关于 x 轴对称的同时仍保持上限为 1。同样，阶跃函数也可以做上述对称变换，得到对称型阶跃函数：输入小于零时其输出为-1，输入大于零时其输出为 1。

MP 模型是最简单的人工神经网络模型，但作为一种理论雏形来说已经足够了。剩下的问题就是如何改善网络拓扑结构和寻求权值学习算法。例如目前流行的多层前馈网络和 BP 算法，就是对网络拓扑结构和权值学习算法的修正。相关的内容我们将在后边结合实例进行具体介绍。

12.1.2　神经网络控制

神经网络用于控制领域，主要是为了解决复杂的非线性、不确定、不确知系统的控制问题。由于神经网络具有模拟人的部分智能的特性，主要是具有学习能力和自适应性，使神经网络控制能对变化的环境具有自适应性，并且成为基本上不依赖于模型的一类控制。因此，神经网络控制已成为“智能控制”的一个新的分支。

截止到目前为止，神经网络在包括 PID 控制在内的控制领域各种模型中都有所应用，其核心算法就是利用神经网络的高度非线性来逼近各种难控模型，再通过一定的校正手段达到控制系统的设计性能指标。

目前常用的有以下几种神经网络控制方式：

- PID 控制。PID 控制是最常用的一种控制方法，也是我们曾经详细分析过的。利用神经网络进行 PID 控制，通过神经控制器（NNC）和神经网络辨识器（NNI）进行参数调整，能够起到智能控制的作用。其结构如图 12-5 所示。

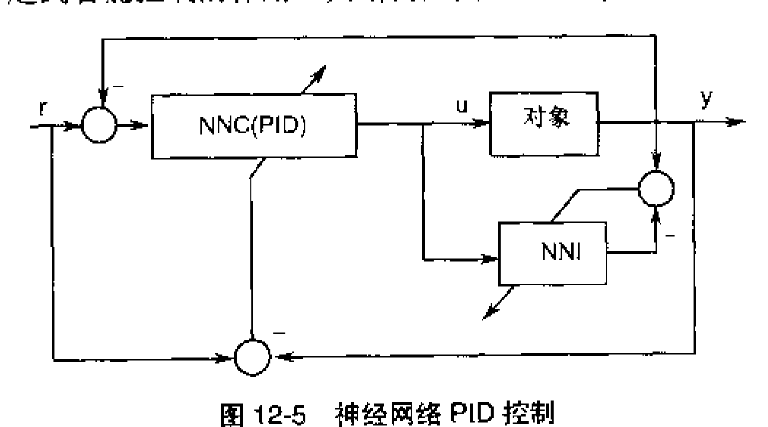


图 12-5　神经网络 PID 控制

- 直接逆动态控制。利用 NNC 与被控对象串联，NNC 实现对象 P 的逆模型 P^{-1}。其优点是能够在线调整模型参数，实现设定值跟踪。
- 间接自校正控制。由 NNI 对被控对象进行在线辨识，根据“确定性等价”原则，设计控制器参数，以达到有效控制的目的。
- 模型参考自适应控制。利用 NNC 和 NNI 构造对象的参考模型，根据神经网络的自调整功能实现在线辨识和控制。
- 内模控制。神经网络内部模型控制（IMC——Internal Model Control），就是首先利用 NNI 对被控对象 P 进行在线辨识，然后用 NNC 实现对象 P 的逆模型 P^{-1}，再加入滤波器提高系统鲁棒性的一种控制方式。其控制器输入由被控对象与内部模型的输出误差来调整。
- 前馈反馈控制。利用 NNC 构造前馈控制器，常规控制器来构造反馈控制器，可以有效的抑制扰动的作用。
- 预测控制。预测控制是一种基于模型的控制，其算法三要素是模型预测、滚动优化和反馈校正。利用神经网络良好的非线性函数逼近性能，可以实现非线性对象的预测模型，从而保证优化目标的实现。

以上神经网络技术在控制领域的一些应用，事实上神经网络的功能远不止如此，下面我们结合 MATLAB 神经网络工具箱具体介绍神经网络的一些用途。

12.2　MATLAB 神经网络工具箱

我们从前边介绍过的最简单的 MP 模型入手，逐步介绍其功能。请看下面这个例子。

在图 12-1MATLAB 神经网络工具箱演示界面下选择“Simple neuron and transfer functions”选项，即可进入如图 12-6 所示的简单神经元 MP 模型演示界面。不过最初进入的演示界面只有一个输入量 p，可以点击图 12-6 界面右下方的第一个按钮“contents”，进入简单神经元模型内容设置界面（图 12-7）。

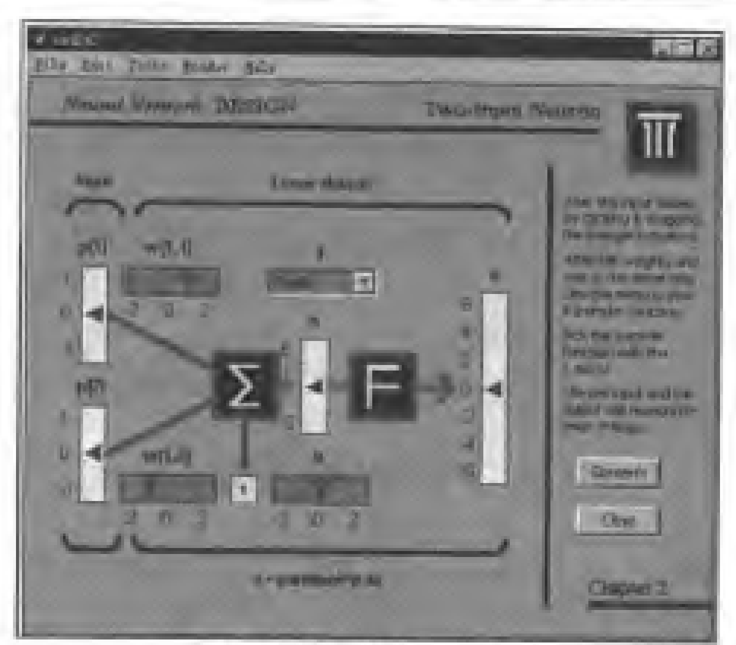


图 12-6　简单神经元 MP 模型



图 12-7　简单神经元模型内容设置界面

该界面包括如下四个选项：

- 神经元模型与网络体系结构（Neuron Model & Network Architectures）；
- 演示实例（An Illustrative Example）；
- 感知器学习规则（Perceptron Learning Rule）；
- 信号与权重向量空间（Signal & Weight Vector Spaces）

我们只需在第一个选项的下拉菜单中选择“Two Input Neuron”选项即可，其余选项均使用 MATLAB 的缺省设置。

下面我们回到图 12-6 所示的 MATLAB 神经网络工具箱演示界面。该界面中间的简单神经元模型与图 12-3 所示的神经网络 MP 模型是完全一致的，只是各个变量字母代表的具体含

义不同。

其中 p（1）和 p（2）代表神经元的两个输入量，也就是图 12-3 中的 y_i；w（1，1）和 w（1，2）代表两个输入量连接到该神经元的权值，也就是图 12-3 中的 w_{ij}；b 是神经元的阈值，也就是图 12-3 中的 θ_i；Σ框代表求和，这一点与图 12-3 是一致的；F 框就是图 12-3 中的非线性作用函数，可以通过该界面下中间标有“F”的下拉菜单中选择具体的函数形式，其菜单格式见图 12-8。a 则表示神经元的输出。



图 12-8　作用函数类型选择

根据上一节的讨论，我们选择对称的 Sigmoid 型作用函数，即从“F”的下拉菜单中选择“logsig”选项。

图 12-6 界面的最下方，还给出了该神经元的仿真模型公式：

$$a = \log sig(w \cdot p + b)$$

确定了输入量、网络拓扑结构和作用函数类型后，我们就可以对该简单神经元网络模型进行仿真了。该演示模型的仿真完全是可视化的，只需用鼠标拖动图中相关参数下边黄色或灰色刻度条里边的红色三角即可改变其参数值（注意，输出量 a 刻度条的黑色三角是不能拖动的，因为这是最终计算的值）。最终的计算结果由 F 框后面的红色箭头显示出来。请看图 12-9 所示的计算结果。

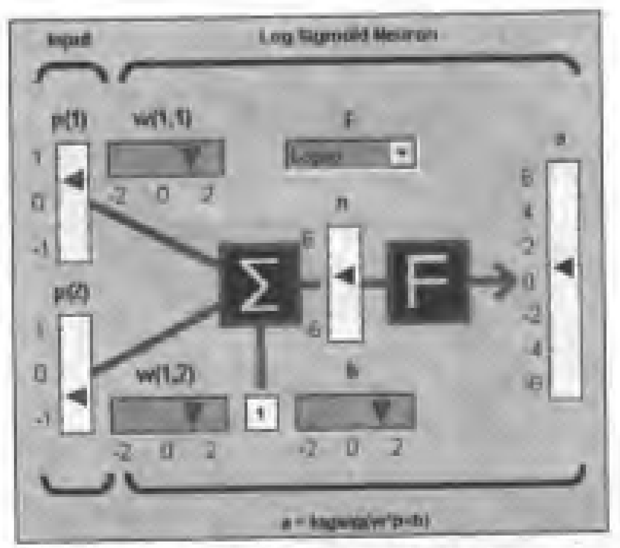


图 12-9　简单神经元计算结果

在图 12-9 中，我们令两个输入量 p（1）=1，p（2）=−1；两个权值 w（1，1）=1，w（1，2）=1；阈值 b=1。从 F 框后面的红色箭头指示的结果来看，该简单神经元的输出结果为 0.8 左右。

小结：本例比较详细的叙述了 MATLAB 环境下如何对简单神经元 MP 模型进行仿真的问题，并根据临时选择的数据给出了仿真结果。该仿真程序秉承了 MATLAB 5.x 可视化的风格，用鼠标即可完成所有的操作，简化了仿真的工作量。

第一次接触神经网络的读者可能会提出这样的疑问：上面这些工作不借助神经网络工具箱，在 MATLAB Command Window 下同样也可以完成，神经网络技术具体有什么作用呢？事实上，上面那个例子是最简单的一个神经元的网络，而实际应用的神经网络中就远大于此了。这么多神经元，没有相应的函数计算，就非常复杂难解了。神经网络正是通过这些庞大的神

经元实现了高度非线性，从而能够任意逼近 L_2 空间中的函数。下面我们首先介绍一些 MATLAB 神经网络工具箱中的函数，然后再通过一个非线性函数逼近的例子具体阐述这些函数的用法。

对于给定的由一组输入向量和输出向量描述的非线性函数，可以首先用 errsurf（）函数来评价其对于单个神经元的非线性程度。该函数的调用格式为：

e= ERRSURF(P,T,WV,BV,F)

其中 P 是 Q 维输入向量，T 是 Q 维输出向量；WV 是单个神经元的输出权值向量，BV 是相应的阈值向量；F 是字符串形式的作用函数。返回值 e 是相对应于不同权值和阈值的误差矩阵。

得到单个神经元的误差矩阵后可以用 plotes（）函数将其以三维图形的方式绘制出来，其调用格式为：

PLOTES(WV,BV,ES,V)

其中 WV 是单个神经元的输出权值向量，BV 是相应的阈值向量；ES 是误差矩阵；V 是图形坐标设置，缺省值为[-37.5, 30]。

在利用神经网络逼近非线性函数时，首先要创建神经网络。常用的是 newlin（）函数，其功能是创建一层线性神经网络，调用格式为：

net=NEWLIN(PR,S,ID,LR)

其中 PR 是 R 个输入向量的最小和最大值矩阵（或向量）；S 是输出向量个数；ID 是输入延迟向量，缺省值为[0]；LR 是学习速率，缺省值为 0.01。返回值 net 就是创建完毕的单层神经网络。

神经网络创建完毕后，可以用 train（）函数进行训练。其调用格式为：

[net,tr] = TRAIN(NET,P,T,Pi,Ai)

其中 NET 就是上一步创建完毕的神经网络，P 是输入向量，T 是期望的非线性输出；Pi 是输入延迟条件，缺省值为 0；Ai 是网络的延迟条件，缺省值为 0。返回值 net 是训练完毕的网络，tr 是训练记录。

神经网络训练完毕后就可以用 sim（）函数检验训练效果并预测新的非线性函数输出了。sim（）函数也可以用于其他模型的仿真，前边在讲述控制系统仿真时曾经介绍过，这里再简要复习一下。其调用格式为：

[T,X,Y] = SIM('model',TIMESPAN,OPTIONS,UT)

其中'model'是模型描述，在这里就是训练好的神经网络 net（可以根据实际意义起另外的名称）；TIMESPAN 是仿真时间间隔；OPTIONS 是仿真参数；UT 是外部输入，也就是用作检验或预测的非线性函数输入。返回值 T 是仿真的时间向量，X 是状态矩阵，Y 就是该仿真模型的输出。

有了这些基本的神经网络仿真函数，我们就可以完成神经网络的一项重要功能——逼近非线性函数的工作了。请看下例。

测得某非线性环节的输入和输出向量如下。试构造一单层神经网络结构逼近此非线性环节，给出逼近的精度，并预测输入为-1.2 时系统的输出。

输入 $P=[1.0\quad 1.5\quad 3.0\quad -1.2]^T$，输出 $T=[0.5\quad 1.1\quad 3.0\quad -1.0]^T$

分析：从输入和输出向量来看，是典型的非线性函数，在控制领域中一般要对其进行线

性化。通常的线性化方法是最小二乘方法（参看本书第1章），需要编制较长的程序，并且精度不容易控制。下面我们根据MATLAB提供的函数，训练单层神经网络逼近这个非线性函数，并给出逼近精度。

结果：输入为-1.2时系统的输出为：

Y =-1.0466

最终的仿真精度为0.0828。

求解过程：本例的求解分为以下几个步骤：

1．绘制原输入和输出向量的误差平面

在MATLAB Command Window下键入下述语句：

```
%输入向量
P=[1,1.5,3,-1.2];
%输出向量
T=[0.5,1.1,3.0,-1.0];
%权值和阈值范围
w_range=-2:0.4:2;
b_range=-2:0.4:2;
%求解误差平面并绘图
ES=errsurf(P,T,w_range,b_range,'purelin');
plotes(w_range,b_range,ES);
```

相应的误差平面图形见图12-10。

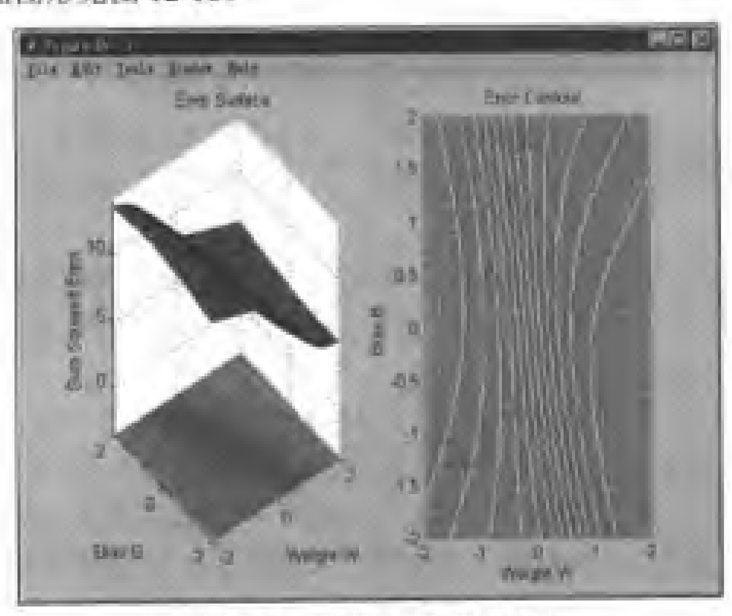


图12-10　非线性函数误差图形

图12-10分为两部分，其中左边是误差平面的三维图形，横纵坐标分别是阈值“Bias”和权值“Weight”，高度坐标是误差平方之和。图中的三维曲面就是利用神经网络仿真时，选择相应的阈值和权值带来的误差平方值。从图中可以读出，这个误差平方值的范围是比较大的，阈值和权值在（-2，2）范围内时（见图中坐标或上文的代码），其值在（5，15）之间。这说明该环节的非线性特性比较严重，用线性化的方法近似不可能完全消除逼近误差。我们的任

务就是利用 MATLAB 提供的神经网络函数，在该误差平面中寻找误差最小的点，并将其作为该非线性环节的神经网络近似。

图 12-10 的右半部分是左边误差平面的辅助曲线，称作误差等高线。其横坐标是权值“Weight”，纵坐标是阈值“Bias”，图形中的白线表示使用该曲线上点的权值和阈值的神经网络拥有相同的误差。

2. 创建并训练神经网络逼近此非线性环节

根据图 12-10 的误差平面和等高线，我们的任务就是寻求误差平面的最低点。MATLAB 提供的神经网络函数可以帮助完成此项功能。紧接上一问的 MATLAB Command Window，键入下述语句：

```
%寻找最快学习速率
maxlr=maxlinlr(P,'bias');
%创建神经网络
net=newlin([-2,2],1,[0],maxlr);
%设定网络训练参数
net.trainParam.epochs = 7;
net.trainParam.goal = 0;
%训练神经网络
[net tr]=train(net,P,T);
%绘制训练轨迹
plotperf(tr,net.trainParam.goal);
grid;
```

上边的代码段中使用了一个新的函数 maxlinlr（），其功能是自动寻找 newlin（）函数创建的神经网络的最快学习速率。其调用格式为：

lr = MAXLINLR(P,'bias')

其中 P 是神经网络的输入向量，'bias'表示包含阈值；lr 是返回的最快学习速率，可以直接被 newlin（）函数调用。

在设定网络训练参数时使用了两个神经网络参数.trainParam.epochs 和.trainParam.goal，分别表示训练的次数和训练目标。一般训练的次数是由实际的问题决定的，因为本例输入和输出数据都较少，所以选择较小的训练次数，否则有可能出现参数发散问题。训练目标可设为 0 或较小的正实数值。

图 12-11 是执行上边代码后得到的训练轨迹。

图 12-11 的横坐标是训练的次数，纵坐标是训练的精度。从图中可以看出训练精度最终趋于 10^{-1} 附近，从曲线的上方也可以读出“Performance is 0.0828293，Goal is 0”，训练结果还是能够令人满意的。事实上，从该神经网络的训练轨迹来看，第一次训练改善最显著，精度从 3 跳变到 0.2 以下，以后的训练基本上都是在 10^{-1} 附近微调。因此，综合时间和训练效果等因素来看，并不是说训练次数越多越好。此时在 MATLAB Command Window 下键入训练记录的变量名“tr”，可以得到训练轨迹的数值：

```
tr =   epoch: [0 1 2 3 4 5 6 7]
       perf: [2.8650 0.1508 0.1284 0.1124 0.1010 0.0928 0.0870 0.0828]
```



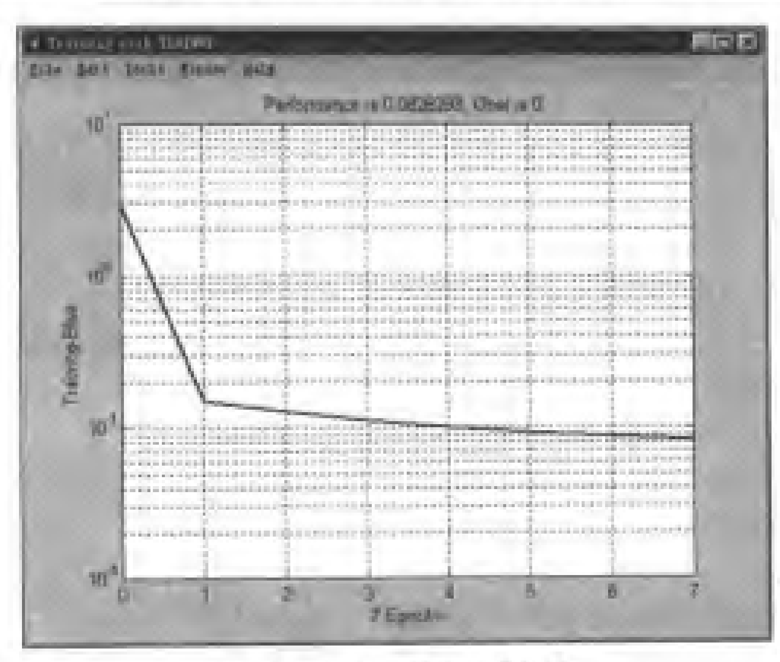


图 12-11　神经网络训练轨迹

3. 校验神经网络逼近结果

神经网络训练结束后，就可以使用该网络模拟此非线性环节了。题目要求是求出输入为-1.2 时系统的输出，可以使用 sim（）函数求得：

```
P=-1.2
Y=sim(net,P);
```

结果为 Y =-1.0466，与期望值-1.0 相差很小，验证了神经网络在逼近非线性函数方面是行之有效的。

此时还可以通过图形来验证神经网络的逼近效果，键入下述语句：

```
plot(P,T,'*',P,Y);
title('neural network approximation');
xlabel('input');
ylabel('output');
grid;
```

绘制的线性神经网络逼近曲线如图 12-12 所示。

图 12-12 中蓝色"*"号是原非线性环节的输入输出点，绿色实线是线性神经网络逼近的函数。从图中可以看出，该线性函数对此非线性环节的逼近是比较理想的。神经网络理论告诉我们，这是最小误差均方意义下的逼近，与最小二乘法得到的结果是基本类似的，仅仅存在非常小的误差。

小结：本例详细介绍了如何在 MATLAB 环境下使用线性神经网络逼近一个非线性环节。从目前的结果来看，除了程序编制比较简单之外还不能看出神经网络与传统的逼近技术相比有什么优点。这一方面是因为本例的数据量比较小，使用传统的逼近技术算法也不复杂；另一方面是因为我们使用的是线性神经网络的缘故。而对于那些数据量非常庞大、结构非常复杂的非线性系统来说，传统的逼近技术算法就有些力不从心，使用多层的非线性神经网络则可以得到很好的结果。

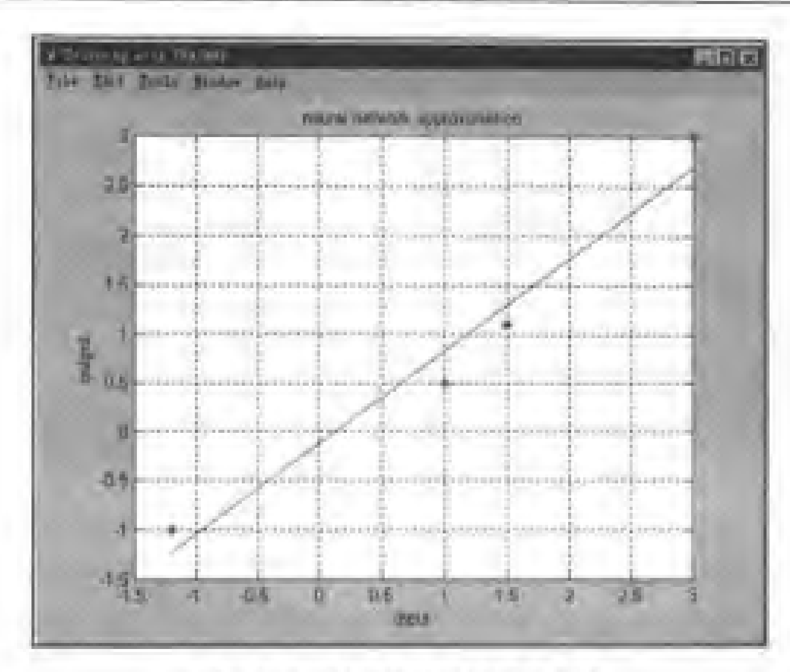


图 12-12 线性神经网络逼近曲线

MATLAB 提供了包括非线性在内的各种神经网络模型，例如目前流行的前馈反向传播网络（Feed－Foreword Back Propagation network），实现函数为 newff（），串级反向传播网络（Cascade－Forward Back Propagation network），实现函数为 newcf（）等等。利用这些函数可以逼近任意复杂程度的 L_2 空间非线性函数。下一节中我们将介绍一些与控制系统有关的常用神经网络模型，需要神经网络工具箱其他相关资料的读者可以查阅 MATLAB 帮助，这里就不再赘述了。

12.3 神经网络控制举例

前边曾经概括地介绍过神经网络技术在控制领域的广泛应用，下面我们就以一个工程实际中的问题为例，具体介绍 MATLAB 环境下神经网络技术在控制领域的应用。

考虑某雷达天线仰角控制系统，通过改变直流电机电流可以控制天线臂角度Φ。系统模型如下：

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ 9.81\sin x_1 - 2x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$$

此处 x_1 是角位移，x_2 是角速度。试利用合适的神经网络模型逼近此非线性系统，并比较相应的控制效果。

结果：

按精确的非线性模型和近似前馈反向传播网络模型控制的阶跃响应曲线如下：

分析：对于本例这种结构比较复杂的完全非线性模型，再仿照上一节的单层线性神经元逼近的方法就很难得到满意的结果了。因此，我们采用较为复杂的前馈反向传播网络，通过多层网络来近似非线性函数。MATLAB 提供了一个用于前馈反向传播网络仿真的 newff（）函数，其调用格式如下：



(a)



(b)

图 12-13　系统的阶跃响应曲线

(a) 精确的非线性模型阶跃响应曲线；(b) 前馈反向传播网络模型阶跃响应曲线。

net = NEWFF(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)

其中 PR 是输入向量的最小最大值矩阵；Si 是第 i 层网络的神经元数；TFi 是第 i 层网络的作用函数，缺省为 Sigmoid 型；BTF、BLF 和 PF 是网络相应的权重学习和训练函数，均可以使用 MATLAB 的缺省设置。

求解过程：

本例的求解分为以下几步：

1．模型变换，分离出非线性函数

这样做一方面是为了控制方便，将线性环节和非线性环节分离，另一方面也是为了更好的利用神经网络逼近非线性函数的特性，使其不干扰线性环节。根据反馈线性化思想，取系统的参考模型为：

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -9x_1 - 6x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 9 \end{pmatrix} r$$

可得系统的非线性反馈环节：$f(X) = 9.81\sin x_1 - 2x_2$

相应的系统的控制规律为：$u = 9r - [9,6]X - f(X)$，其中 r 为设定值参考输入。

2．利用前馈反向传播网络逼近系统的非线性环节

分离出系统的非线性环节以后，就可以利用神经网络对其进行逼近了。由系统的模型可知，所要设计训练的神经网络具有两个输入 x_1 和 x_2。网络设计目的是能够使其产生逼近函数 f（x）的非线性输出，所以输出层只有一个输出神经元。中间选取一个两层神经元，Sigmoid 型采用作用函数，输出层取线性作用函数， 在 MATLAB Command Window 下键入网络的源代码：

```
%系统状态变量初始化
x1=[rands(1,300)*pi,rands(1,100)*pi];
x2=[rands(1,300)*pi,zeros(1,100)*pi];
P=[x1;x2];
%离散化时间间隔
dt=0.05;
```




```
%理想非线性输出
T=x2+dt*[9.81*sin(x1)-2*x2];
[R,Q]=size(P);
%第一层神经元数目
S1=10;
%第二层神经元数目
[S2,Q]=size(T);
pr=[0.7,1.05;0.95,1.04];
%初始化网络
net=newff(pr,[S1 S2])
xx1=[(1:300)/100];
xx2=[zeros(1,300)];
xx=[xx1;xx2];
y=xx2+dt*[9.81*sin(xx1) -2*xx2];
%循环次数
net.trainParam.epochs = 50;
%训练精度
net.trainParam.goal = 0;
%网络训练
[net,tr] = train(net,xx,y);
plotperf(tr,net.trainParam.goal);
grid;
```

在上边的代码中，我们采用包含两个隐层的前馈反向传播网络，第一层神经元数目为 10，第二层神经元数目为 1。同时取神经网络的训练次数为 50 次，要求的训练精度的目标为 0。此时相应的训练轨迹如图 12-14 所示。

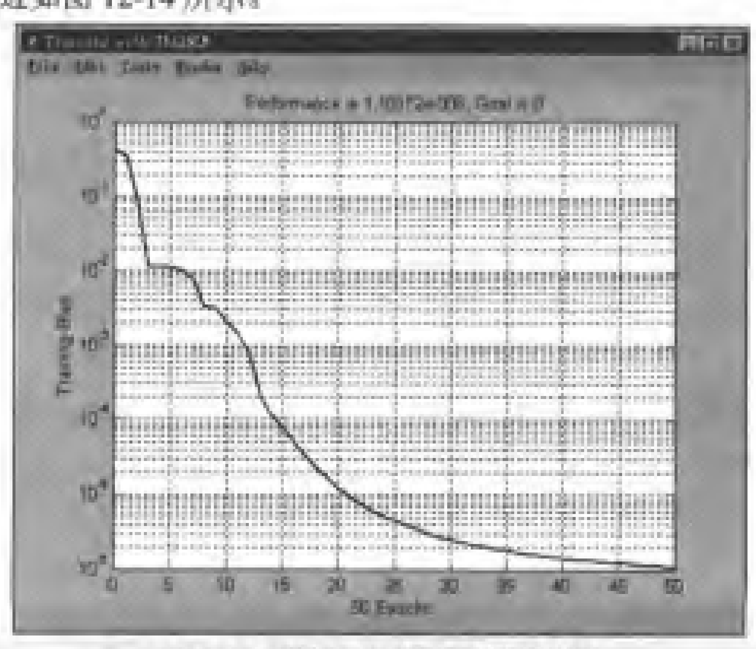


图 12-14　前馈反向传播网络训练轨迹

从图中可以读出，训练 50 次以后达到的训练精度为 1.10172e -6，远低于上一节单层线性神经元的 10^{-1}，这也是我们采用前馈反向传播网络的原因。可以预测，在这样的训练精度下，该神经网络对系统非线性环节的逼近一定是非常好的。下面我们通过一定的数据验证这个说法。在 MATLAB Command Window 下键入如下代码：

```
%数据初始化，取 60 个点
xx1=[(1:30)/10];
xx2=[zeros(1,30)];
xx=[xx1;xx2];
y=xx2+dt*[9.81*sin(xx1)-2*xx2];
%模型仿真
y2 = sim(net,xx);
%图形输出
plot(xx,y,'*',xx,y2);
title('neural network approximation');
xlabel('input');
ylabel('output');
grid;
```

此时系统的仿真曲线如图 12-15 所示。

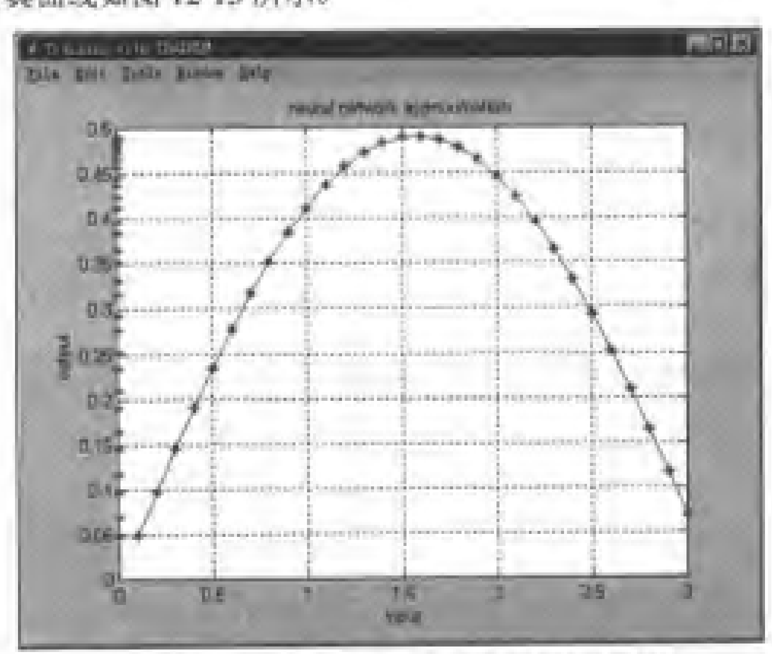


图 12-15　前馈反向传播网络逼近非线性函数轨迹

图中蓝色的“*”号是非线性函数的输出，红色的实线是神经网络的逼近曲线。可以看出，两条轨迹几乎完全重合，可以认为此时的神经网络已经完全实现了所要逼近的非线性函数的功能。事实上，前边已经给出训练精度为 10^{-6} 数量级，在这样的精度下肉眼是很难判断两条曲线之间的差别的。

3．控制效果仿真及对比

有了训练好的神经网络模型之后，就可以仿真评价其控制效果了。首先在 SIMULINK 下



搭建原系统的方框图如图 12-16 所示。该方框图中包括三个"Sum"求和元件、两个"State Space"系统状态空间描述元件、两个"Fcn"函数元件、一个"Step"阶跃输出元件、一个"Scope"示波器元件、一个"Matrix Gain"矩阵增益元件和一个"Mux"多路开关元件。对此方框图进行仿真，可以得到如结果中图 12-13 系统的阶跃响应曲线（a）。然后就可以去掉图 12-16 中的两个"Fcn"函数元件，加入一个"MATLAB Fcn"函数元件，在该元件的设置对话框中填入代码：

```
y=sim(net,u)
```

对此方框图进行仿真，可以得到如图 12-13 系统的阶跃响应曲线（b）。

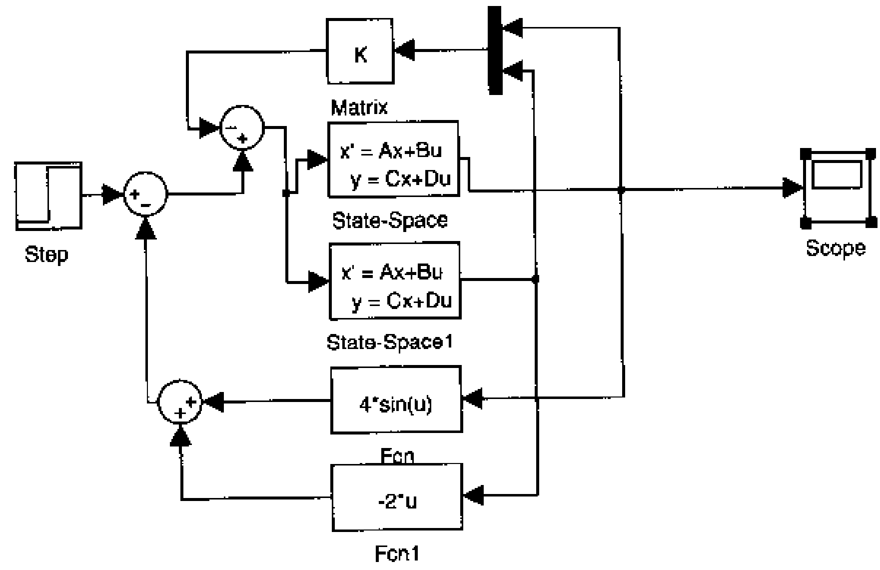


图 12-16　非线性系统仿真方框图

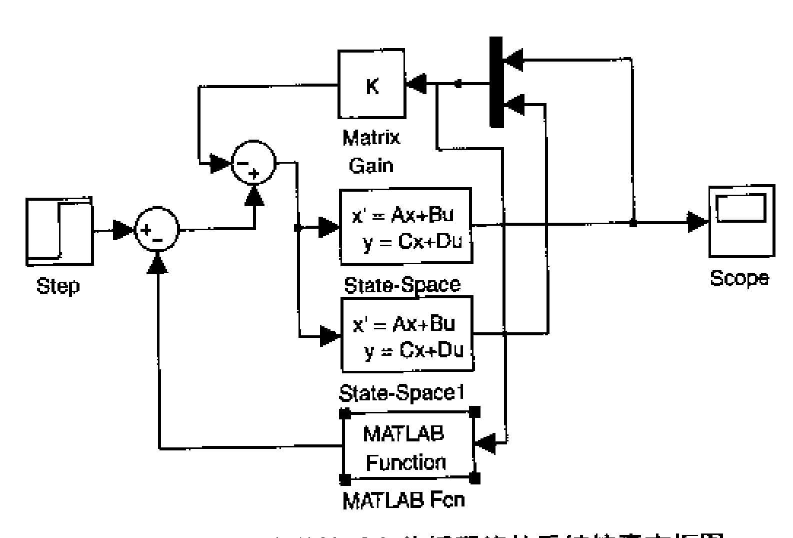


图 12-17　包含前馈反向传播网络的系统仿真方框图

对比这两幅阶跃响应曲线，可以看出曲线的形状和变化规律都基本一致，仅在细微的地方有所差别（注意两幅曲线的坐标刻度不完全一样）。这说明利用此前馈反向传播网络进行控制的效果是可以满意的。

小结：本例详细介绍了前馈反向传播网络在控制系统中的一个应用实例，给出了完整的 MATLAB 解法。实际运行了上边代码的读者会发现，利用包含前馈反向传播网络的方框图进

行仿真比包含非线性环节的方框图仿真慢很多，这是因为每步仿真都要调用该神经网络，而神经网络的节点较多，对于计算机来说，显然不如简单的正弦函数计算简便。当然，相对于控制效果来说，这点开销是可以忍受的。

本章我们首先介绍了一些神经网络理论的基本概念及其在控制领域的应用，然后给出了MATLAB 神经网络工具箱中一些常用的函数，最后在此基础上解决了一个实际的控制问题。简单地说，神经网络的核心就是利用简单的结点和复杂的拓扑结构逼近任意阶次的非线性函数。在控制领域，尤其是时延和非线性程度较高的流程工业控制问题中，利用神经网络进行控制也是当前的热点之一。

总之，神经网络在控制系统中的应用前景还是非常广阔的，MATLAB 强大的神经网络工具箱也一定会逐步更新、扩展。

参 考 文 献

1 方崇智，萧德云. 过程辨识. 北京：清华大学出版社，1988

2 Ljung L. System Identification - Theory for the User, Prentice Hall, Upper Saddle River, N.J. 2nd edition, 1999.

3 郑大钟. 线性系统理论. 北京：清华大学出版社，1995

4 Kailath, T., Linear Systems, Prentice-Hall, 1980

5 冯纯伯，田玉平，忻欣. 鲁棒控制系统设计. 南京：东南大学出版社，1995

6 舒迪前. 预测控制系统及应用. 北京：机械工业出版社，1996

7 张文修，梁广锡. 模糊控制与系统. 西安：西安交通大学出版社，1998

8 厉玉鸣. 非线性控制系统. 北京：化学工业出版社，1985

9 Lennart Ljung，For Use with MATLAB System Identification Toolbox，Mathworks Inc,2001

10 For Use with MATLAB Control System Toolbox, Mathworks Inc,2001

11 For Use with MATLAB Robust Control Toolbox, Mathworks Inc,2001

12 Manfred Morari,N. Lawrence Ricker, For Use with MATLAB Model Predictive Control Toolbox, Mathworks Inc,2001

13 For Use with MATLAB Fuzzy Logic Toolbox, Mathworks Inc,2001

14 For Use with MATLAB Nonlinear Control Design Blockset, Mathworks Inc,2001